Gradient-based optimization of flow through porous media

Jan-Dirk Jansen

Version 3, September 2016
Abstract
These lecture notes form part of the course material for the MSc course AES1490 *Advanced Reservoir Simulation* which has been taught at TU Delft over the past decade as part of the track *Petroleum Engineering and Geosciences* in the two-year MSc program *Applied Earth Sciences*.

The notes cover the gradient-based optimization of subsurface flow. In particular they treat optimization methods in which the gradient information is obtained with the aid of the adjoint method, which is, in essence, an efficient numerical implementation of implicit differentiation in a multivariate setting.

Chapter 1 reviews the basic concepts of multivariate optimization and demonstrates the equivalence of the Lagrange multiplier method for constrained optimization and the use of implicit differentiation to obtain gradients in the presence of constraints.

Chapter 2 introduces the use of Lagrange multipliers and implicit differentiation for the optimization of large-scale numerical systems with the adjoint method. In particular it addresses the optimization of oil recovery from subsurface reservoirs represented as reservoir simulation models, i.e. space- and time-discretized numerical representations of the nonlinear partial differential equations that govern multi-phase flow through porous media. It also covers the use of robust adjoint-based optimization to cope with the inherent uncertainty in subsurface flow models and addresses some numerical implementation aspects.

Chapter 3 gives a brief overview of various further topics related to gradient-based optimization of subsurface flow, such as closed-loop reservoir management and hierarchical optimization of short-term and long term reservoir performance.
Contents

1 OPTIMIZATION THEORY ........................................................................................................7
  1.1 Introduction ...................................................................................................................7
  1.2 Notation .......................................................................................................................7
  1.3 Unconstrained optimization ..........................................................................................8
     1.3.1 Optimality conditions .............................................................................................8
     1.3.2 Convexity ...............................................................................................................10
  1.4 Constrained optimization ............................................................................................11
     1.4.1 Single equality constraint ....................................................................................11
     1.4.2 Lagrange multipliers ...........................................................................................14
     1.4.3 Multiple equality constraints ................................................................................18
     1.4.4 Interpretation of the Lagrange multipliers ............................................................20
     1.4.5 Inequality constraints ............................................................................................22
  1.5 Numerical optimization ...............................................................................................26
     1.5.1 Gradient-based and gradient-free methods ............................................................26
     1.5.2 Search direction .....................................................................................................27
     1.5.3 Step size ................................................................................................................29
  1.6 References for Chapter 1 .............................................................................................29

2 ADJOINT-BASED FLOODING OPTIMIZATION ..........................................................31
  2.1 Introduction ..................................................................................................................31
  2.2 Problem statement .......................................................................................................32
     2.2.1 System model ........................................................................................................32
     2.2.2 Objective function ................................................................................................33
     2.2.3 Constraints .............................................................................................................33
  2.3 Optimal control theory ................................................................................................34
     2.3.1 Adjoint equation - derivation ...............................................................................34
     2.3.2 Lagrangian and Hamiltonian* ..............................................................................37
     2.3.3 Adjoint equation – interpretations* .........................................................................38
  2.4 Constrained optimization ............................................................................................42
     2.4.1 Input constraints and output constraints ...............................................................42
     2.4.2 Bound constraints on the input ..............................................................................43
     2.4.3 External constraint handling ..................................................................................43
     2.4.4 Internal constraint handling ...................................................................................44
  2.5 Simple example ..........................................................................................................47
  2.6 Robust optimization ....................................................................................................49
     2.6.1 Multiple realizations ..............................................................................................49
1 Optimization theory

1.1 Introduction

This chapter reviews some aspects of optimization theory as a precursor to Chapter 2, in which we will address techniques for reservoir flooding optimization, or life-cycle optimization. This concerns the optimization of the recovery factor of a reservoir, or another economic objective, through manipulation of input variables such as water injection rates or bottom hole pressures over the producing life of the reservoir. Many numerical techniques are available to solve optimization problems. An important distinction is between methods that attempt to find a global optimum, and those that can find a local optimum only. For realistic problems, all optimization techniques involve some form of iteration, and the ‘local methods’ will produce answers that are dependent on the initial guess used as starting point for the iteration. Another distinction is between gradient-based and gradient-free methods. Gradient-based methods make use of gradients, i.e. derivatives of the optimization objective with respect to the input variables, to guide the iteration process. Gradients of a function have the property that they point in the direction of maximum increase of the function value, which explains their significance to find the maximum (or the minimum) of a function.‡ A disadvantage of gradient-based methods is that they usually converge to a local optimum, as opposed to some gradient-free techniques that can search for the global optimum. However, gradient-free methods require many more function evaluations (i.e. reservoir simulations) than gradient-based methods to find an optimum, which makes them unattractive for our purpose. In the following we will discuss some concepts from optimization theory. In particular we will state the optimality conditions for constrained and unconstrained optimization of a function of multiple variables, and we will review the method of Lagrange multipliers. There is a large amount of literature available treating optimization problems at widely varying levels of mathematical complexity. Accessible texts aimed at practical applications are Gill et al. (1986), Fletcher (1987), Rao (1996), and Luenberger and Ye (2010). Numerical aspects are covered extensively in Bonnans et al. (2003) and Nocedal and Wright (2006), which also provide an in-depth treatment of the theoretical aspects. For the theory covered in the current section on optimization theory we have in particular relied on Gill et al. (1986), and we refer to that text for further details and proofs.

1.2 Notation

Scalars will be indicated with Latin or Greek, lower or upper case letters, and vectors with Latin or Greek lower case letters in bold-face or in index notation. Occasionally we will use a sans-serif font to indicate vectors with a special meaning. Matrices are indicated with Latin or Greek bold-face capitals. The superscript $T$ is used to indicate the transpose, and dots above variables to indicate differentiation with respect to time. A colon is sometimes used in subscripts to indicate a range of variables, e.g. $u_{1:K}$ represents $u_k$, $k = 1, 2, \ldots, K$. Unless indicated otherwise, vectors are always considered to be column vectors. E.g. a vector $u \in \mathbb{R}^n$ is defined as

‡ Finding the minimum of a function is identical to finding the maximum of minus one times that function.
\begin{equation}
\mathbf{u} \triangleq \begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
\nu_n
\end{bmatrix}.
\end{equation}

This expression also illustrates the use of the ‘embellished’ equality sign $\triangleq$ to introduce definitions. We use the ‘equivalence sign’ $\equiv$ instead of the ordinary equality sign $=$ to make a distinction between role of the equality signs in an equation. E.g., in the equation

\begin{equation}
a \equiv b = 0,
\end{equation}

the equivalence sign just indicates that the variables $a$ and $b$ represent the same quantity but expressed in different forms, while the equality sign expresses the requirement that the quantity should be equal to zero, i.e. a condition that needs to be fulfilled.

1.3 Unconstrained optimization

1.3.1 Optimality conditions

Consider the unconstrained optimization problem

\begin{equation}
\min_u \mathcal{J}(u),
\end{equation}

or in words “minimize the objective function $\mathcal{J}(u)$ by changing the control variable $u$”. Here $\mathcal{J}$ is a univariate function, i.e. a function of a single variable which is $u$ in this case. The conditions for a minimum $u = u^0$ are taught in any basic calculus course and we will briefly review them as a precursor to more complex problems. The first-order and second-order necessary conditions for a minimum are given by* 

\begin{equation}
\frac{\partial \mathcal{J}}{\partial u}
\bigg|_{u=u^0} = 0,
\end{equation}

\begin{equation}
\frac{\partial^2 \mathcal{J}}{\partial u^2}
\bigg|_{u=u^0} \geq 0,
\end{equation}

while the first-order and second-order sufficient conditions are given by condition (1.4) together with

\begin{equation}
\frac{\partial^2 \mathcal{J}}{\partial u^2}
\bigg|_{u=u^0} > 0.
\end{equation}

More in general, a point $u = u^0$ that satisfies the necessary condition (1.4) is called a stationary point or a critical point$^{\dagger}$. It is a minimum if the sign of the second derivative is positive (as in equation (1.6)), and a maximum if the sign is negative. In both cases, the point is called an extreme or an optimal point, and the necessary and sufficient conditions for an optimal point are therefore also known as optimality conditions. If the second derivative is equal to zero, $u^0$ is either an inflection point or an extreme depending on the sign of the

$^{\dagger}$ The objective function is sometimes referred to as a performance function, or a cost function, which is then to be maximized or minimized respectively.

* We tacitly assume that $\mathcal{J}$ is continuous and at least twice differentiable.

$^{\ddagger}$ We use superscripts 0 to indicate stationary points.
higher derivatives. Conditions (1.4) to (1.6) can be derived through approximating $J$ in the neighborhood of $u^0$ using a Taylor expansion:

$$J(u) = J(u^0) + \left[ \frac{\partial J(u)}{\partial u} \right]_{u=u^0} (u-u^0) + \frac{1}{2} \left[ \frac{\partial^2 J(u)}{\partial u^2} \right]_{u=u^0} (u-u^0)^2 + \ldots .$$  \hspace{1cm} (1.7)

The necessary condition (1.4) can be obtained by considering the first two terms of this expansion: for any non-zero value of the first-order derivative it would be possible to find a smaller value for $J$ than $J(u^0)$. Next, by considering also the second-order term, and assuming that the first necessary condition is fulfilled, it follows that the second necessary condition is given by inequality (1.5) because for any other choice of the second-order derivative the function value $J(u)$ would decrease in the neighborhood of $u^0$. Because inequality (1.5) still allows for the possibility that $J(u)$ either increases or stays constant, we need the strict inequality (1.6) to obtain the sufficient condition: for any other choice the function value $J(u)$ would not increase in the neighborhood of $u^0$. A similar set of conditions can be obtained for the case where $J$ is a multivariate function of variables $u_1, u_2, \ldots, u_m$, corresponding to an optimization problem

$$\min_u J(u), \; u = [u_1 \; u_2 \; \ldots \; u_m]^T.$$ \hspace{1cm} (1.8)

The first-order necessary condition then becomes

$$\left. \frac{\partial J}{\partial u} \right|_{u=u^0} = \left[ \frac{\partial J}{\partial u_1} \; \frac{\partial J}{\partial u_2} \; \ldots \; \frac{\partial J}{\partial u_m} \right]_{u=u^0} = 0^T.$$ \hspace{1cm} (1.9)

Alternatively, this condition can be expressed as

$$\nabla J|_{u=u^0} = \left[ \frac{\partial J}{\partial u_1} \; \frac{\partial J}{\partial u_2} \; \ldots \; \frac{\partial J}{\partial u_m} \right]_{u=u^0} = 0,$$ \hspace{1cm} (1.10)

where we introduced the gradient vector $\nabla J$ (or, simply, gradient) which, by definition, is equal to the transpose of the derivative vector: $\nabla J = (\partial J / \partial u)^T$. The second-order necessary condition is the requirement that the Hessian matrix

$$H \triangleq \left. \frac{\partial^2 J}{\partial u^2} \right|_{u=u^0} \equiv \left[ \begin{array}{cccc} \frac{\partial^2 J}{\partial u_1^2} & \frac{\partial^2 J}{\partial u_1 \partial u_2} & \ldots & \frac{\partial^2 J}{\partial u_1 \partial u_m} \\ \frac{\partial^2 J}{\partial u_2 \partial u_1} & \frac{\partial^2 J}{\partial u_2^2} & \ldots & \frac{\partial^2 J}{\partial u_2 \partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u_m \partial u_1} & \frac{\partial^2 J}{\partial u_m \partial u_2} & \ldots & \frac{\partial^2 J}{\partial u_m^2} \end{array} \right]_{u=u^0} \geq 0.$$ \hspace{1cm} (1.11)
where the inequality sign implies that the matrix is positive semi-definite, i.e. that all its eigenvalues are larger than or equal to zero. The sufficient conditions are given by condition (1.9) (or 1.10) together with the requirement that

$$\frac{\partial^2 J}{\partial u^2} > 0$$

(1.12)

where the strict inequality sign now implies that the matrix is positive definite, i.e. that all its eigenvalues are larger than zero. In parallel to the univariate case, a point $u = u^0$ that satisfies the first-order necessary condition (1.9) is called a stationary point. Depending on the sign of the eigenvalues of the Hessian (1.12) it can be a minimum (all eigenvalues larger than zero), a maximum (all eigenvalues smaller than zero) or a saddle point (some eigenvalues larger, some smaller than zero). If one or more of the eigenvalues are equal to zero, and the remaining ones have an equal sign, it will be required to consider the higher derivatives to determine the character of the stationary point.

1.3.2 Convexity

In some circumstances the necessary conditions are also sufficient conditions. In particular this is often the case if the optimization problem is convex. A convex set $S$ is defined as a collection of points such that a line connecting any two points of the set is entirely within the set. If the set consists of points $s \in \mathbb{R}^m$ this can be written as

$$\text{if } s_1, s_2 \in S \text{ then } s \in S,$$

(1.13)

where

$$s = \beta s_1 + (1-\beta)s_2, 0 \leq \beta \leq 1.$$  

(1.14)

Figure 1.1 gives some examples of two-dimensional convex and non-convex sets. A convex function $J(u): \mathbb{R}^m \rightarrow \mathbb{R}$ is defined as a function for which the epigraph, i.e. all points $s \in \mathbb{R}^{m+1}$ above the graph of $J(u)$, form a convex set. For smooth, twice differentiable functions $J(u): \mathbb{R} \rightarrow \mathbb{R}$, convexity is equal to the requirement that the second derivative $d^2 J/du^2$, which is also known as the curvature, is positive everywhere. The concepts of convexity and positive curvature can be extended to smooth functions $J(u): \mathbb{R}^m \rightarrow \mathbb{R}$, in which case it is required that the Hessian matrix $\partial^2 J/\partial u^2$ is positive definite for all values of $u$. Figure 1.2 gives some examples of one-dimensional convex and non-convex functions, and illustrates that for smooth convex problems the necessary conditions for an optimum are often also sufficient conditions. An unconstrained optimization problem is called convex if the epigraph of the objective function is a convex set. In constrained optimization problems the constraints typically restrict the domain of the optimization variables and thus also the epigraph. Constraints can therefore make an optimization problem nonconvex, despite convexity of the objective function. Practical optimization problems in reservoir engineering are seldom convex. However, the concept of convexity is an essential element of optimization theory and forms an important ingredient in the derivation of many optimization algorithms.
Figure 1.1: Convex and nonconvex sets of two-dimensional points.

Figure 1.2: Convex and nonconvex functions $J(u): \mathbb{R} \rightarrow \mathbb{R}$. The figures show only parts of the graphs for $u \in [-1,1]$ and $J(u) \in [0,1]$. The dots represent stationary points. The solid dots represent minima (i.e. optimal points for a minimization problem). For the convex function $J(u) = 0.1 + u^2$ (left figure) the single stationary point is automatically the optimal point.

1.4 Constrained optimization

1.4.1 Single equality constraint

Simple example

As an introduction to constrained optimization we will discuss a simple example involving a single equality constraint. First, consider the unconstrained optimization problem

$$\min_u J(u), \quad u = [u_1, u_2]^T,$$

where

$$J(u) = 2(u_1^2 + u_2^2).$$

Equation (1.16) represents a paraboloid, and it can be seen immediately that the minimum is given by $u^0 = 0$; see Figure 1.3 (top left). Formally we find this result by setting the derivatives of $J$ with respect to $u_1$ and $u_2$ equal to zero, i.e.
\[
\frac{\partial J}{\partial u} = \begin{bmatrix}
\frac{\partial J}{\partial u_1} & \frac{\partial J}{\partial u_2}
\end{bmatrix} = \begin{bmatrix}
4u_1 & 4u_2
\end{bmatrix} = 0^T,
\] (1.17)

and solving for the stationary point \( u^0 \) from the two resulting equations \( 4u_1 = 0 \) and \( 4u_2 = 0 \). In that case we have automatically satisfied the first-order necessary condition (1.9). Next we take the second derivatives, to check if they fulfill the second-order necessary condition (1.11), or, more interestingly, the second-order sufficient condition (1.12). The first step of this formal procedure can also be expressed as setting the total differential \( \delta J \) of the function \( J \) equal to zero:

\[
\delta J \equiv \frac{\delta J}{\delta u} \delta u = \begin{bmatrix}
4u_1 & 4u_2
\end{bmatrix} \begin{bmatrix}
\delta u_1 \\
\delta u_2
\end{bmatrix} = 0.
\] (1.18)

The differentials \( \delta J \) and \( \delta u \) are often referred to as the variations of \( J \) and \( u \) respectively\(^1\). The two equations for \( u_1 \) and \( u_2 \) follow by realizing that equation (1.18) should hold for arbitrary values of the variations \( \delta u_1 \) and \( \delta u_2 \) and therefore that \( 4u_1 = 0 \) and \( 4u_2 = 0 \). The second-order conditions can be verified similarly by taking the second variation:

\[
\delta^2 J \equiv \delta u^T \frac{\delta^2 J}{\delta u^2} \delta u = \begin{bmatrix}
\delta u_1 \\
\delta u_2
\end{bmatrix} \begin{bmatrix}
4 & 0 \\
0 & 4
\end{bmatrix} \begin{bmatrix}
\delta u_1 \\
\delta u_2
\end{bmatrix}.
\] (1.19)

Because the Hessian at the right-hand side of equation (1.19) is diagonal its eigenvalues are equal to the diagonal terms, and because these are both larger than zero the matrix is positive definite and \( u^0 \) is indeed a minimum\(^8\).

**Elimination of the constraint**

Next, consider a constrained optimization example, given by the same problem statement (1.15), but now under the equality constraint

\[
c(u) = u_1 + u_2 - 0.6 = 0.
\] (1.20)

Equation (1.20) represents a line in the \( u_1 - u_2 \) plane passing through \( (0, 0.6) \) and \( (0.6, 0) \); see Figure 1.3 (top right and bottom right). The solution to this constrained optimization problem can be obtained by solving for \( u_1 \) or \( u_2 \) from equation (1.20), substituting the result in equation (1.16), and then proceeding as in the unconstrained case, but now taking derivatives only to the one remaining variable:

---

\(^1\) Sometimes the use of the word variation is restricted to differentials of a functional, i.e. of a function of a function.

\(^8\) Note that we have introduced three different notations for the first-order conditions: 1) \( \nabla J/\delta u = 0^T \), 2) \( \nabla J = 0 \), and 3) \( \delta J = 0 \). The three analogous notations for the second-order necessary conditions are: 1) \( \nabla^2 J/\delta u^2 \geq 0 \), 2) \( \nabla^2 J \geq 0 \), and 3) \( \delta^2 J \geq 0 \), where \( \nabla^2 J \equiv H^T = (\nabla^2 J/\delta u^2)^T \).
Figure 1.3: Constrained optimization. Top left: contour lines of the objective function $\mathcal{J}(\mathbf{u})$. Top right: constraint $c(\mathbf{u})$ in the $(u_1,u_2)$ plane. Bottom left: Contour lines of the objective function intersected by a vertical plane (represented by a set of horizontal lines) through the constraint. The constrained minimum is at the bottom of the parabola in this plane. Bottom right: top view displaying the projected contour lines of the objective function together with the constraint. The dot indicates the minimum, and coincides with the point where the constraint curve is tangent to the contour lines.
\[ u_2 = 0.6 - u_1, \] (1.21)
\[ \mathcal{J} = 4u_1^2 - 2.4u_1 + 0.72, \] (1.22)
\[ \delta \mathcal{J} \equiv \frac{\partial \mathcal{J}}{\partial u_1} \delta u_1 = (8u_1 - 2.4)\delta u_1. \] (1.23)

The first-order necessary condition for a minimum is obtained by requiring that \( \delta \mathcal{J} = 0 \) for arbitrary values of \( \delta u_1 \), which results in the solution \( u_1^0 = u_2^0 = 0.3 \), corresponding to a value of the objective function \( \mathcal{J}^0 = 0.36 \). The second-order sufficient condition is obtained by requiring that \( \delta^2 \mathcal{J} > 0 \) for arbitrary values of \( \delta^2 u_1 \), and because
\[ \delta^2 \mathcal{J} \equiv \frac{\partial^2 \mathcal{J}}{\partial u_1^2} \delta u_1^2 = 8\delta u_1^2, \] (1.24)
the solution is indeed a minimum. It is the lowest point of the parabola given by equation (1.22), as can also be seen in Figure 1.3 (bottom left).

**1.4.2 Lagrange multipliers**

*Modified objective function*

For more complex constraints it may not be possible to explicitly solve for one of the input variables as we could do in this simple example. In that case, the classic way of solving the constrained problem is through the use of the *Lagrange multiplier* technique. This involves the definition of a *modified objective function*:\footnote{In texts on constrained optimization the modified objective function is often referred to as the *Lagrangian*. Here we will restrict the use of the term Lagrangian for a quantity that plays a role in dynamic optimization, as will be discussed in Chapter 2.}
\[ \bar{\mathcal{J}}(u, \lambda) \triangleq \mathcal{J}(u) + \lambda e(u) \equiv 2(u_1^2 + u_2^2) + \lambda (u_i + u_2 - 0.6), \] (1.25)

where the constraint equation has been added to the original objective function after multiplication with the Lagrange multiplier \( \lambda \). Note that if the constraint equation is satisfied, the term containing \( \lambda \) becomes zero, such that the value of the modified objective \( \bar{\mathcal{J}} \) becomes equal to value of the original objective \( \mathcal{J} \). We will discuss the meaning of the Lagrange multiplier later, and for the time being we just consider \( \lambda \) as an additional variable such that \( \bar{\mathcal{J}} \) is now a function of three variables: \( u_1, u_2 \) and \( \lambda \). The first step of the minimization procedure, taking the total differential and setting the result equal to zero, then gives
\[ \delta \bar{\mathcal{J}} \equiv \frac{\partial \bar{\mathcal{J}}}{\partial u} \delta u + \frac{\partial \bar{\mathcal{J}}}{\partial \lambda} \delta \lambda \]
\[ \equiv [4u_1 + \lambda - 4u_2 + \lambda] [\delta u_1] + (u_1 + u_2 - 0.6) \delta \lambda = 0. \] (1.26)

The first-order necessary condition for a minimum requires that equation (1.26) should hold for arbitrary values of the variations \( \delta u_1, \delta u_2 \), and \( \delta \lambda \), which results in three equations
\[ \frac{\partial \bar{\mathcal{J}}}{\partial u_1} \equiv 4u_1 + \lambda = 0, \] (1.27)
\begin{align}
\frac{\partial J}{\partial u_2} &= 4u_2 + \lambda = 0 \quad \text{(1.28)} \\
\frac{\partial J}{\partial \lambda} &= u_1 + u_2 - 0.6 = 0 \quad \text{(1.29)}
\end{align}

from which we can solve for the three variables to obtain

\[
\begin{bmatrix}
u^0 \\ \lambda^0
\end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.3 \\ -1.2
\end{bmatrix},
\]

which is indeed the same result for \( u^0 \) as was found before.

**Directional derivative**

As discussed above, and as illustrated in Figure 1.3 (bottom left), the minimum of the constrained problem occurs at the bottom of the parabola that is formed by only considering those values of the objective function corresponding to points \((u_1, u_2)\) that obey the constraint equation (i.e. those values that are exactly above the constraint ‘curve’ (which here is a line) in the \((u_1, u_2)\) plane). We could therefore also search for the minimum by considering the directional derivative of the objective function along the constraint which is defined as the length of the vector formed by projecting the derivative \( \frac{\partial J}{\partial u} \) on the tangent to the constraint curve:

\[
\frac{\partial J}{\partial u} \bigg|_v \triangleq \frac{\partial J}{\partial u} v = \left[ \frac{\partial J}{\partial u_1} \frac{\partial J}{\partial u_2} \right] v^	op,
\]

where \( v \equiv v(u) \) is the unit vector of the tangent in a point \((u_1, u_2)\). Filling in the numbers from the numerical example and equating the result to zero gives

\[
\frac{\partial J}{\partial u} \bigg|_v \equiv \begin{bmatrix} 4u_1 & 4u_2 \\ u_1 & u_2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \equiv 4u_1 - 4u_2 = 0,
\]

which, together with constraint equation (1.20) results in the same optimum that we found before. Note that the directional derivative becomes zero when \( v \) is perpendicular to \( \frac{\partial J}{\partial u} \). In that case \( v \) is not only tangent to the constraint but also tangent to the contour lines of the objective function; see Figure 1.3 (bottom right). At the same time \( \frac{\partial J}{\partial u} \) will then have the same or the opposite direction as the derivative \( \frac{\partial c}{\partial u} \). Therefore we find that at the minimum we have

\[\text{Points } (u_1, u_2) \text{ obeying the constraints are known as } \text{feasible points}, \text{ and the set of all feasible points as the feasible set.}\]

\[\text{In our simple example the components } [1 \quad -1] \text{ of } v \text{ are constant for all values of } u_1 \text{ and } u_2; \text{ see Figure 1.3 (top right and bottom right). However, in the general case } v \text{ is a function of } u.\]

\[\text{The directional derivative is the transpose of the } \text{projected gradient of } J, \text{ defined as } \nabla J |_v \triangleq v^\top \nabla J.\]

\[\text{The constraint equation } c(u) = 0 \text{ can be interpreted as a single contour line of the function } c(u). \text{ Just like for the objective function, the steepest descent direction, i.e. the maximum value of } \frac{\partial c}{\partial u} \text{, is perpendicular to the contour lines, and therefore to the constraint curve in the } (u_1, u_2) \text{ plane.}\]
\[ \frac{\partial J}{\partial u} \bigg|_{u_{\text{in}}} = -\lambda \frac{\partial c}{\partial u} \bigg|_{u_{\text{in}}}, \]  

where \( \lambda \) is an arbitrary constant. Equation (1.33) implies that the two derivative vectors have the same or opposite directions, but different magnitudes with a ratio that is equal to the arbitrary constant \( \lambda \). The reason to choose \( \lambda \) to be preceded by a minus sign becomes clear if we differentiate definition (1.25) with respect to \( u \) which results in

\[ \frac{\partial \bar{J}}{\partial u} = \frac{\partial J}{\partial u} + \lambda \frac{\partial c}{\partial u}, \]

and because we have \( \frac{\partial \bar{J}}{\partial u} = 0 \) we find that for the optimal value \( u^0 \) equations (1.33) and (1.34) are now identical. The arbitrary constant \( \lambda \) is therefore just the Lagrange multiplier that was introduced in a more ad-hoc way before. The use of the minus sign serves only to remain consistent with our earlier definition of the Lagrange multiplier. Returning to equation (1.31) we can simply verify that the first-order necessary condition is satisfied, i.e. that the directional derivative

\[ \begin{bmatrix} \frac{\partial \bar{J}}{\partial u_1} \\ \frac{\partial \bar{J}}{\partial u_2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 4u_1 \\ 4u_2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \]

is indeed equal to zero in the stationary point \( u^0 = [0.3 \\ 0.3]^T \).

**Implicit differentiation**

An alternative route to introduce the Lagrange multiplier method is as follows. A key step in the elimination of the constraint as initially considered in the simple example above was the possibility to write \( u_2 \) as an explicit function of \( u_1 \); see equation (1.21). For more complex constraints it will usually not be possible to compute such an explicit relationship. However, it will often be possible to compute the derivative \( \frac{\partial u_2}{\partial u_1} \) and in that case we can simply write

\[ \delta \bar{J} = \left( \frac{\partial \bar{J}}{\partial u_1} + \frac{\partial \bar{J}}{\partial u_2} \frac{\partial u_2}{\partial u_1} \right) \delta u_1. \]

The key step in this procedure is the computation of the derivative \( \frac{\partial u_2}{\partial u_1} \). Usually this can not be done explicitly, in which case we can use the technique of *implicit differentiation* as follows. Starting from the implicit constraint equation

\[ c(u_1, u_2) = 0, \]

we find that

\[ \delta c = \frac{\partial c}{\partial u_1} \delta u_1 + \frac{\partial c}{\partial u_2} \delta u_2 = 0, \]

which leads to

\[ ^1 \text{Alternatively, some texts define the modified objective function as } J(u, \lambda) = J(u) - \lambda c(u), \text{ i.e. with a minus sign in front of the Lagrange multiplier } \lambda. \]
Gradient-based optimization of flow through porous media, version 3, September 2016

\[ \delta u_2 = -\left( \frac{\partial c}{\partial u_2} \right)^{-1} \frac{\partial c}{\partial u_1} \delta u_1, \] (1.39)

and thus to the implicit derivative

\[ \frac{\partial u_2}{\partial u_1} = -\left( \frac{\partial c}{\partial u_2} \right)^{-1} \frac{\partial c}{\partial u_1}. \] (1.40)

Substitution of this expression in equation (1.36) results in

\[ \delta J = \left[ \frac{\partial J}{\partial u_1} - \frac{\partial J}{\partial u_2} \left( \frac{\partial c}{\partial u_2} \right)^{-1} \frac{\partial c}{\partial u_1} \right] \delta u_1. \] (1.41)

In a stationary point this expression should be zero for arbitrary values of \( \delta u_1 \) which means that

\[ \frac{\partial J}{\partial u_1} + \lambda_1 \left( \frac{\partial c}{\partial u_2} \right)^{-1} \frac{\partial c}{\partial u_1} = 0, \] (1.42)

where we have now introduced \( \lambda_1 \) as a short notation for \(-\partial J / \partial u_2 (\partial c / \partial u_2)^{-1}\). Following a similar reasoning we can derive that

\[ \frac{\partial J}{\partial u_2} + \lambda_2 \left( \frac{\partial c}{\partial u_2} \right)^{-1} \frac{\partial c}{\partial u_2} = 0, \] (1.43)

where \( \lambda_2 \) is a short notation for \(-\partial J / \partial u_1 (\partial c / \partial u_1)^{-1}\). Now, if \( \lambda_1 = \lambda_2 \), i.e. if

\[ \frac{\partial J}{\partial u_2} \left( \frac{\partial c}{\partial u_2} \right)^{-1} = \frac{\partial J}{\partial u_1} \left( \frac{\partial c}{\partial u_1} \right)^{-1}, \] (1.44)

we can combine equations (1.42) and (1.43) into

\[ \frac{\partial J}{\partial u} + \lambda \frac{\partial c}{\partial u} = 0. \] (1.45)

Condition (1.44) implies that the components of \( \partial J / \partial u \) and \( \partial c / \partial u \) are pair-wise proportional which is just the case in a stationary point. Comparison of equation (1.45) with equation (1.33) shows that in this way we again find \( \lambda \) as an arbitrary constant multiplying the derivative of the constraint. Filling in the numerical values of the example in equation (1.42) and in the constraint (1.20) we obtain

\[ \begin{align*}
4u_i^0 + \frac{-4u_i^0 \times (1)}{\lambda} \times 1 &= 0, \\
u_i^0 + u_i^0 - 0.6 &= 0
\end{align*} \]

which is indeed again the result derived before, and which illustrates that the use of Lagrange multipliers to compute the derivatives of a constrained objective function can also be
interpreted as a form of implicit differentiation. An essential role in this derivation is played by equation (1.38) which defines the admissible variations, i.e. those combinations of variations $\delta u_1$ and $\delta u_2$ that keep the constraint condition fulfilled.

### 1.4.3 Multiple equality constraints

#### Feasible arcs

Until now we considered the minimization of a function with a single equality constraint, but the method of Lagrange multipliers can be generalized to cope with multiple constraints which may be equalities, inequalities or a combination of both. We will discuss inequality constraints later, while we will now address the extension to multiple equality constraints $c_i = 0, i = 1, \ldots, q$, which can be stacked in a constraint vector $c$ as

$$c(u) = 0, \quad c \in \mathbb{R}^q, \quad u \in \mathbb{R}^m. \tag{1.47}$$

In general, the number of constraints, $q$, should be less than or equal to the number of control variables, $m$, otherwise the problem is over-constrained and there will be no solution except for special cases. This requirement can be made more precise in case of multiple linear constraints, which can be expressed as

$$c(u) = Au + b = 0. \tag{1.48}$$

where $A \in \mathbb{R}^{q \times m}$ and $b \in \mathbb{R}^q$. If the rows of $A$ are independent, each row of the matrix-vector equation (1.48) removes a degree of freedom from the optimization problem. The number of independent rows of $A$, i.e. its rank, therefore determines the number of constraints. In the extreme case of $\text{rank}(A) = m$, we have as many constraints as input variables $u_i, i = 1, \ldots, m$, and there is no freedom left for optimization. Another property that follows from equation (1.48) is the condition for admissible variations. Because

$$\frac{\partial c}{\partial u} = A, \tag{1.49}$$

we find, in analogy to equation (1.38), that admissible variations in case of multiple linear constraints have to obey

$$A \delta u = 0. \tag{1.50}$$

I.e., the admissible variations are in the null space of the matrix $A$. Points $u$ that obey the constraint equations are called feasible points. The set of all feasible points $u$ is called the feasible set, and the admissible variations $\delta u$ the feasible directions. Note that any movement in a feasible direction will remain on the constraint. In case of nonlinear constraints we can use a Taylor expansion around a feasible point $u^*$,

$$c(u) = c(u^*) + \frac{\partial c}{\partial u_{u^*}} \bigg|_{u_u^*} (u - u^*) + \ldots, \tag{1.51}$$

such that $A$ can be interpreted as

$$A = A(u^*) = \frac{\partial c}{\partial u_{u^*}}. \tag{1.52}$$

In this nonlinear case it is not sufficient to just consider feasible directions, i.e. straight lines along which we can move without violating the constraints, but we have to consider feasible
arcs, i.e. curved lines. Equation (1.50) is therefore still a necessary condition for admissible variations, but no longer a sufficient one. For example, consider a three-dimensional optimization problem with input vector \( \mathbf{u} = [u_1, u_2, u_3]^T \) and two nonlinear constraints

\[
\begin{align*}
  c_1 &\equiv u_3 - (u_1^2 + u_2^2) = 0 \\
  c_2 &\equiv u_3 + (u_1^2 + u_2^2) = 0.
\end{align*}
\]

The constraints are paraboloids that just touch each other in the origin which is therefore the only feasible point. The matrix \( \mathbf{A} \) in the feasible point follows as

\[
\mathbf{A} = \left. \frac{\partial \mathbf{c}}{\partial \mathbf{u}} \right|_{\mathbf{u} = \mathbf{0}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix},
\]

such that any vector \( \mathbf{u} = [\delta u_1, \delta u_2, 0]^T \) fulfills equation (1.50), i.e. the null space is identical to the \( u_1-u_2 \) plane. However, because the origin is the only feasible point, feasible arcs simply don’t exist in this case. The additional requirements that are necessary to specify admissible variations in case of nonlinear constraints are known as constraint qualifications. These exist in various forms, see e.g. Bonnans et al. (2003) or Nocedal and Wright (2006), but most of them are outside the scope of our text. An exception is the simple constraint qualification given by the requirement that \( \mathbf{A} \) has full row rank, i.e. that it has independent rows. Consider a three-dimensional optimization problem again, but this time with the constraints

\[
\begin{align*}
  c_1 &\equiv u_3 - u_1 = 0 \\
  c_2 &\equiv u_3 - u_1^2 = 0.
\end{align*}
\]

These constraints are a plane and a parabolic cylinder (a ‘parabolic tunnel’) respectively that intersect each other at two lines: one for which \( u_1 = u_3 = 0 \) (i.e. the \( u_2 \)-axis) and a parallel one for which \( u_1 = u_3 = 1 \). The matrices \( \mathbf{A} \) now become

\[
\begin{align*}
\mathbf{A}_1 &\equiv \left. \frac{\partial \mathbf{c}}{\partial \mathbf{u}} \right|_{\mathbf{u} = [0, u_1, 0]^T} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \\
\mathbf{A}_2 &\equiv \left. \frac{\partial \mathbf{c}}{\partial \mathbf{u}} \right|_{\mathbf{u} = [1, u_1, 0]^T} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 1 \end{bmatrix},
\end{align*}
\]

which both have independent rows and therefore meet the constraint qualification. The admissible variations are in the null spaces of \( \mathbf{A}_{1,2} \) and are given by \( \delta \mathbf{u} = [0, \delta u_2, 0]^T \). In other words the feasible arcs are in this case lines with feasible directions parallel to the \( u_2 \)-axis.

**Necessary optimality conditions**

Under the assumption that \( \mathbf{A} \) has full row rank, i.e. that the constraint qualifications are met, we can now derive the first-order necessary conditions for a minimum in case of multiple constraints by considering the modified objective function (c.f. equation (1.25))

\[
\begin{align*}
\tilde{\mathcal{J}}(\mathbf{u}, \lambda) &\equiv \mathcal{J}(\mathbf{u}) + \lambda^T \mathbf{c}(\mathbf{u}),
\end{align*}
\]

where \( \lambda \in \mathbb{R}^q \) is a vector of Lagrange multipliers. Just as for the single-constraint case, stationarity of the modified objective function provides the first-order necessary conditions for a constrained minimum. The complete set of necessary first-order optimality conditions can therefore be written as

Gradient-based optimization of flow through porous media, version 3, September 2016
\[
\frac{\partial J}{\partial \mathbf{u}} = 0^T, \\
\frac{\partial J}{\partial \lambda} = 0^T, 
\]

(1.61)

(1.62)

where equation (1.62) is just a restatement of the equality constraint (1.47). Conditions (1.61) and (1.62) are sometimes referred to as the Euler-Lagrange equations. As before we may, alternatively, consider the directional derivatives along the constraints, and, in analogy to equation (1.31), write

\[
\left. \frac{\partial J}{\partial \mathbf{u}} \right|_V \triangleq \frac{\partial J}{\partial \mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} \begin{bmatrix} v_1 & v_2 & \cdots & v_p \end{bmatrix},
\]

(1.63)

where \( v_i \equiv v(u)_i, i=1,\ldots,q \) are the unit vectors of feasible directions, tangent to the constraints, in a feasible point \( u \). In fact, because the admissible variations along the constraints obey equation (1.50), the columns of \( V \) may be any set of vectors that form a basis for the null space of \( A \). Following the same reasoning as for the single-constraint case we note that we obtain \( \partial J/\partial \mathbf{u}|_{V=0^T} \) when all columns \( v \) of \( V \) are perpendicular to \( \partial J/\partial \mathbf{u} \). In that case the columns \( v \) are not only tangent to the corresponding constraints \( c_i \) but also tangent to the contour lines of the objective function, and the derivatives \( \partial c_i/\partial \mathbf{u} \) will then have the same or the opposite direction as \( \partial J/\partial \mathbf{u} \):

\[
\left. \frac{\partial J}{\partial \mathbf{u}} \right|_{u=u^0} = -\lambda^T \frac{\partial c}{\partial \mathbf{u}} \bigg|_{u=u^0} \equiv -\lambda^T A^0,
\]

(1.64)

which implies that in a stationary feasible point \( u^0 \) the derivative of the objective function, \( \partial J/\partial \mathbf{u} \), is a linear combination, with coefficients \( \lambda_i \), of the constraint derivatives \( \partial c_i/\partial \mathbf{u}, i=1,\ldots,q \). Note that, just as in equation (1.33) we added a minus sign in front of \( \lambda \) to remain consistent with our earlier definition of the Lagrange multipliers.

1.4.4 Interpretation of the Lagrange multipliers

The magnitude of the Lagrange multipliers can be interpreted as a measure of the effect of perturbing the constraints on the value of the objective function. This can be seen by applying definition (1.60) to a perturbed constraint \( \tilde{c}(\mathbf{u}) = c(\mathbf{u}) + \delta c \):

\[
\tilde{J}(\mathbf{u},\lambda) = J(\mathbf{u}) + \lambda^T \tilde{c}(\mathbf{u}) \equiv J(\mathbf{u}) + \lambda^T \left[ c(\mathbf{u}) + \delta c \right].
\]

(1.65)

The difference in the modified objective function value between a perturbed and an unperturbed constraint then follows as

\[
\delta \tilde{J} = \tilde{J}(\mathbf{u},\lambda) - J(\mathbf{u},\lambda) = \lambda^T \delta c.
\]

(1.66)

In the perturbed and unperturbed optima we have \( \tilde{J}(\mathbf{u}^0,\lambda^0) = J(\mathbf{u}^0) \) and \( \tilde{J}(\tilde{\mathbf{u}}^0,\tilde{\lambda}^0) = \tilde{J}(\tilde{\mathbf{u}}^0) \), and therefore we can also write

\[
\delta \tilde{J} \equiv \tilde{J}(\mathbf{u}^0) - J(\mathbf{u}^0) = \lambda^T \delta c.
\]

(1.67)

\[1. \quad \text{The perturbation } \delta c \text{ as defined here can also be interpreted as the residual in the constraint, i.e. its deviation from } 0.\]
As an illustration, consider a spherical objective function in three-dimensional input space with its center in \((u_1, u_2, u_3) = (2, 0, 0)\):

\[
\mathcal{J}(u) = (u_1 - 2)^2 + u_2^2 + u_3^2,
\]

and with constraints given by equations (1.56) and (1.57) such that the modified objective function becomes

\[
\mathcal{J}(u, \lambda) = \mathcal{J}(u) + \lambda^T c = (u_1 - 2)^2 + u_2^2 + u_3^2 + [\lambda_1 \lambda_2 \lambda_3] \begin{bmatrix} u_3 - u_1 \\ u_3 - u_2 \end{bmatrix}.
\]

Proceeding in the usual fashion, i.e. taking the first variation and setting the coefficients multiplying \(\delta u_i\) up to \(\delta \lambda_2\) equal to zero, we obtain the set of five equations

\[
\frac{\partial \mathcal{J}}{\partial u_1} = 2(1- \lambda_2)u_1 - (4 + \lambda_1) = 0, \tag{1.70}
\]

\[
\frac{\partial \mathcal{J}}{\partial u_2} = 2u_2 = 0, \tag{1.71}
\]

\[
\frac{\partial \mathcal{J}}{\partial u_3} = 2u_3 + \lambda_1 + \lambda_2 = 0, \tag{1.72}
\]

\[
\frac{\partial \mathcal{J}}{\partial \lambda_1} = u_3 - u_1 = 0, \tag{1.73}
\]

\[
\frac{\partial \mathcal{J}}{\partial \lambda_2} = u_3 - u_2^2 = 0, \tag{1.74}
\]

from which the constrained minimum and the corresponding objective function value can be computed as either

\[
\begin{bmatrix} u_1^0 & u_2^0 & u_3^0 & \lambda_1^0 & \lambda_2^0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -4 & 4 \end{bmatrix} \quad \text{with} \quad \mathcal{J}(u^0) = 4, \tag{1.75, 1.76}
\]

or

\[
\begin{bmatrix} u_1^0 & u_2^0 & u_3^0 & \lambda_1^0 & \lambda_2^0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & -2 & 0 \end{bmatrix} \quad \text{with} \quad \mathcal{J}(u^0) = 2, \tag{1.77, 1.78}
\]

such that clearly the latter solution corresponds to the minimum; see also Figure 1.4 (left). If we specify the perturbed constraints

\[
\tilde{c}_1 = u_1 - u_1 + \delta c_1 \quad \text{and} \quad \tilde{c}_2 = u_3 - u_1^2 + \delta c_2.
\]

we can repeat the computation for the minimum which leads to

\[
\tilde{u}^0 \equiv \begin{bmatrix} \tilde{u}_1^0 & \tilde{u}_2^0 & \tilde{u}_3^0 \end{bmatrix}^T \approx \begin{bmatrix} 1 - \delta c_1 + \delta c_2 & 0 & 1 - 2 \delta c_1 + \delta c_2 \end{bmatrix}^T, \tag{1.81}
\]

and

\[
\mathcal{J}(\tilde{u}^0) \approx 2 - 2 \delta c_1, \tag{1.82}
\]
where we used the Taylor expansion $\sqrt{1+\varepsilon} = 1 + \frac{1}{2}\varepsilon - \ldots$ and neglected the quadratic terms in $\delta c_1$ and $\delta c_2$; see Figure 1.4 (right). Clearly, the magnitude of the Lagrange multipliers ($\lambda_1^0 = -2$ and $\lambda_2^0 = 0$) is a first-order measure of the effect on $J$ of perturbing the constraints with $\delta c_1$ and $\delta c_2$. In fact perturbing $c_2$ while staying on $c_1$ does not have any effect at all (at least to first order) because the corresponding Lagrange multiplier $\lambda_2^0$ is equal to zero. This can also be understood by considering Figure 1.4: staying in the plane of the paper and moving away from the optimal point $u^0$ (the solid dot) along the linear constraint means staying tangent to the circles, at least up to first order. In other words, in this case we do not need the second constraint to arrive at the constrained minimum $u^0$.

![Figure 1.4: Spherical objective function with two constraints. Left: Cross-section in the $u_1$-$u_3$ plane. The circles are cross-sections through the contour spheres of the objective function. The line and the parabola are cross-sections through the plane and the parabolic cylinder that form the constraints $c_1$ and $c_2$. The two dots are cross-sections through the two lines that together form the feasible set. The dots also indicate the constrained stationary points where these two feasible lines (which are perpendicular to the paper) are tangent to the contour spheres of the objective function. The solid dot corresponds to the minimum. Right: Detail showing the effect of perturbing the constraints. The solid dot corresponds to the minimum in the unperturbed case. Open dots nr. 1 and nr. 2 correspond to the minima when perturbing constraint $c_1$ and $c_2$ with $\delta c_1 = 0.15$ and $\delta c_2 = 0.15$ respectively, while open dot nr. 3 corresponds to the minimum when perturbing both constraints.](image)

1.4.5 Inequality constraints

Constraint activity

Once we allow the possibility of inequality constraints, we need to make a distinction between feasible points that are on a constraint, and those that are not. The constraints related to these two categories of feasible points are called active constraints and inactive constraints respectively. Both categories of feasible points are said to satisfy the constraints, whereas infeasible points violate the constraints. Inactive constraints do not restrict the feasibility of perturbations from a feasible point, i.e., even if a feasible point $u$ is very close to a constraint,
there is, in theory, always\footnote{In numerical computations, this may be an impractically small value such that we can consider the constraint active for practical purposes, but for the time being we will restrict our attention to the theoretical case.} room for a small feasible change $\delta u$. Active constraints, however, restrict the feasible perturbations. Two types of feasible perturbations can now be distinguished: those that keep the constraint active, known as \textit{binding perturbations}, and those that make it inactive by moving away from it, known as \textit{non-binding perturbations}. To derive the first- and second-order optimality conditions we have to consider the active constraints. Starting with a set of linear inequality constraints

$$d(u) = Au + b \leq 0 \tag{1.83}$$

where $d \in \mathbb{R}^q$ and $u \in \mathbb{R}^m$, we can partition them as

$$\begin{bmatrix} \hat{d} \\ \hat{a} \end{bmatrix} = \begin{bmatrix} \hat{A} \\ \hat{X} \end{bmatrix} u + \begin{bmatrix} \hat{b} \\ \hat{y} \end{bmatrix} \leq 0 \tag{1.84}$$

where the hatted and the striked-through coefficients $\hat{A}$ and $\hat{b}$ correspond to the active and inactive constraints $\hat{d}$ and $\hat{a}$ respectively. In analogy to the equality-constrained case we can now derive that admissible variations in the form of \textit{binding perturbations} are in the null space of $\hat{A}$:

$$\hat{A} \delta u = 0. \tag{1.85}$$

Assuming that the rows of $\hat{A}$ are linearly independent, i.e. that the constraint qualification is met, a first-order necessary condition for optimality of a feasible point with active linear constraints $\hat{d}$ is therefore given by (c.f. equation (1.64))

$$\frac{\partial J}{\partial u} \bigg|_{u=\hat{u}^0} = -\hat{\lambda}^T \frac{\partial \hat{d}}{\partial u} \bigg|_{u=\hat{u}^0} = -\hat{\lambda}^T \hat{A} 0, \tag{1.86}$$

where $\hat{\lambda}$ is the vector of Lagrange multipliers corresponding to the active constraints. In analogy to the case with pure equality constraints, equation (1.86) implies that in a stationary feasible point $\hat{u}^0$ the derivative of the objective function, $\frac{\partial J}{\partial u}$, is a linear combination, with coefficients $\hat{\lambda}$, of the active constraint derivatives $\frac{\partial \hat{d}_i}{\partial u}$, $i = 1, \ldots, \hat{r}$. However, we should also consider the possibility of variations in the form of \textit{non-binding perturbations}, in which case it holds for at least one row $\hat{a}^i$ of $\hat{A}$ (corresponding to active constraint $i$) that

$$\hat{a}^i \delta u < 0, \tag{1.87}$$

or that

$$\hat{a}^i \delta u > 0. \tag{1.88}$$

Inequality (1.87) corresponds to a variation $\delta u$ around the feasible point $u$ that results in moving off the constraint into a feasible direction because in that case

$$\hat{a}^i (u + \delta u) + b = \hat{a}^i u + b = 0, \tag{1.89}$$

such that inequality (1.83) remains valid, but with a constraint that now becomes inactive. Oppositely, inequality (1.88) corresponds to moving off the constraint into an infeasible direction, i.e. to violating the constraint. In order for a feasible point $u$ that obeys equation...
(1.86) to be a minimum, all non-binding perturbations $\delta u$ should first of all be in a feasible direction, i.e. they should obey equation (1.87). For all perturbations we therefore require that
\[ \hat{A}^0 \delta u \leq 0. \] (1.90)
Moreover, the non-binding perturbations should result in an increase of $J$, because if at least one of them would result in a decrease, clearly $u$ would not be a minimum. In other words we require that for any $\delta u$ obeying equation (1.90) we also have
\[ \frac{\partial J}{\partial u_{lu-u^*}} \delta u \geq 0. \] (1.91)
With the aid of equation (1.86) this can be rewritten as
\[ \hat{\lambda}^T \hat{A}^0 \delta u \leq 0, \] (1.92)
and therefore, since $\delta u$ obeys equation (1.90), we find the additional necessary condition
\[ \hat{\lambda} \geq 0. \] (1.93)
A similar reasoning holds for the case of nonlinear inequality constraints
\[ d(u) \leq 0. \] (1.94)
Using a Taylor expansion around a feasible point $u^*$,
\[ d(u) = d(u^*) + \frac{\partial d}{\partial u_{lu-u^*}} (u - u^*) + \ldots, \] (1.95)
and introducing the partitioning $d = [d^T \ d^T]^T$ in active and inactive constraints as before, the matrix $\hat{A}$ now becomes a function of $u^*$:
\[ \hat{A} = \hat{A}(u^*) \triangleq \left. \frac{\partial d}{\partial u_{lu-u^*}} \right|_{u^*}. \] (1.96)
As discussed in the previous section, the magnitude of the Lagrange multipliers is a first-order measure of the effect of perturbing the constraints. The special case of equation (1.93) in which at least one multipliers is zero implies a situation where the corresponding constraint, although active, is actually not functioning (at least to first order) because slightly moving the constraint does not change the objective function value; see also the example in Figure 1.4 (right). Such an active constraint with a zero Lagrange multiplier is said to be weakly active, as opposed to an active constraint with a positive multiplier which is called strongly active. Most texts on optimization also introduce Lagrange multipliers corresponding to inactive constraints which are then given a zero value by definition.

**Necessary optimality conditions**
Just as for the equality-constrained case we can now define the modified objective function (c.f. equation (1.60))
\[ \tilde{J}(u, \lambda) \triangleq J(u) + \lambda^T d(u), \] (1.97)
and express the first-order necessary conditions for a minimum as
\[ \frac{\partial \mathcal{J}}{\partial \mathbf{u}} = \mathbf{0}^T, \]
\[ \frac{\partial \mathcal{J}}{\partial \lambda} \leq \mathbf{0}^T, \]
\[ \lambda \geq \mathbf{0}^T, \]
\[ \lambda^T \mathbf{d} = 0. \]

Equations (1.98) to (1.101) are referred to as the Karush-Kuhn-Tucker (KKT) conditions, or sometimes just the Kuhn-Tucker conditions, and they can be interpreted as a special form of the Euler-Lagrange equations, adapted to inequality constraints. Note that equation (1.99) is just the constraint condition (1.94). Equation (1.101) is called the complementarity condition. For the case where either \( \lambda_i \) or \( d_i \) are zero for each of the constraints \( i = 1, 2, \ldots, r \), but never both, the term strict complementarity is used. As an example of an inequality-constrained optimization problem, consider the three-dimensional case with a spherical objective function (1.68) that was considered above, but now with a single inequality constraint

\[ d = u_2^2 - u_1 \leq 0. \]

Assuming that the constraint is active, the modified objective function is

\[ \mathcal{J}(\mathbf{u}) \equiv \mathcal{J}(\mathbf{u}) + \lambda d = (u_1^2 - 2)^2 + u_2^2 + u_3^2 + \lambda (u_1^2 - u_3). \]

Taking the derivative with respect to \( \mathbf{u} \) leads to the four equations
\[ \frac{\partial \mathcal{J}}{\partial u_1} = 2(1 + \lambda)u_1 - 4 = 0, \]
\[ \frac{\partial \mathcal{J}}{\partial u_2} = 2u_2 = 0, \]
\[ \frac{\partial \mathcal{J}}{\partial u_3} = 2u_3 - \lambda = 0, \]
\[ \frac{\partial \mathcal{J}}{\partial \lambda} = u_1^2 - u_3 = 0, \]

which can be solved to give
\[ \begin{bmatrix} u_1^0 & u_2^0 & u_3^0 & \lambda^0 \end{bmatrix}^T \approx \begin{bmatrix} 0.84 & 0 & 0.70 & 1.39 \end{bmatrix}^T \text{ and } \mathcal{J}(\mathbf{u}^0) \approx 1.84. \]

The Lagrange multiplier is nonzero and positive, which implies that the inequality constraint is indeed strongly active; see also Figure 1.5. In the more general case of multiple inequality constraints it is usually necessary to perform a trial and error procedure to establish which constraints are active, and many search strategies have been developed for that purpose, for a description of which we refer to the literature listed in Section 1.1.

\[ \uparrow \] The solution for \( u_1^0 \) follows trivially from the second equation. The other three equations can be combined to give a cubic equation for \( u_1^0 \) which is most easily solved numerically, e.g. using the MATLAB command \texttt{roots}.
Gradient-based optimization of flow through porous media, version 3, September 2016

1.5 Numerical optimization

1.5.1 Gradient-based and gradient-free methods

The optimization problems in the previous section could mostly be solved directly. This involved computation of the first variation of the objective function in closed form, setting the result equal to zero, and then solving for the optimal input variables. For large scale optimization problem, like the ones we encounter in reservoir simulation, this is almost never possible, and we need a numerical, usually iterative, procedure to compute the optimum. Many numerical optimization methods exist and we refer again to the textbooks of Gill et al. (1986), Fletcher (1987), Rao (1996), Bonnans et al. (2003) or Nocedal and Wright (2006) for overviews.

A major distinction between the different methods is obtained by dividing them in those that use gradient information, and those that do not. The classic metaphor for maximizing an objective function is climbing to the top of a hill while being surrounded by fog. The simplest conceivable algorithm to automate that activity is the steepest ascent method, where we take steps towards the top such that every step points in steepest upward direction; see Figure 1.6. (Most optimization texts consider minimization instead of maximization, in which case the method is, of course, called the steepest descent method.) This illustrates the two important elements in gradient-based numerical optimization: determining the search direction and determining the step size at each step. A key feature of gradient-based methods is their tendency to find a local optimum, rather than the global one, because once they have reached the top of a hill there is usually no mechanism to let them step towards another, higher top in the landscape. Searching for a global optimum with a gradient-based method therefore requires additional features such as e.g. starting from many initial guesses, and occasionally randomly perturbing the search direction. This is as opposed to most gradient-free methods which directly aim at finding the global optimum, usually also with some form of random
sampling of the objective function value. The price to pay for searching for the global optimum, whichever method used, is the need to perform many function evaluations, i.e. computations of the objective function value. In most optimization problems in reservoir simulation a function evaluation is equivalent to performing a full simulation and the use of global search techniques is therefore computationally very expensive. Because the number of function evaluations used in gradient-free methods is typically an order of magnitude larger than the number used in gradient-based ones, the application of gradient-free methods in reservoir flooding optimization is restricted to cases with a small number of input parameters, say up to a few tens, unless massively parallel computing is used; see Echeverria et al. (2010). In this text we will therefore only address gradient-based methods. Moreover, we will only address some basic aspects and we refer to the vast literature on numerical optimization (see, e.g., the text books listed above) for more in-depth treatments and alternative methods.

Figure 1.6: Contour lines of an objective function with two steepest ascent approaches to the maximum using two different, fixed, step sizes. The solid line represents an effective ascent (the maximum is in this case, with some luck, reached exactly) which is also quite efficient (the path is going reasonably directly to the top); the dotted line represents a less effective ascent (the maximum is never reached exactly) which is also not very efficient (the trajectory ‘zig zags’).

1.5.2 Search direction

Newton-Raphson

Consider an unconstrained optimization problem with a convex objective function $J(u)$. As discussed in Section 1.3.1, the necessary first-order condition for an optimum is

$$\frac{\partial J}{\partial u} \bigg|_{u^*} = 0^T.$$

(1.110)

Suppose that we are in a point $u^*$ away from the optimum $u^0$. Equation (1.110) can then be approximated in $u^*$ with the aid of a first-order Taylor expansion:

$$0^T \approx \frac{\partial J}{\partial u} \bigg|_{u^0-u^*} + (u^0-u^*)^T \frac{\partial^2 J}{\partial u^2} \bigg|_{u^0-u^*}.$$

(1.111)
From this expression we can derive that

\[
(u^0)^T = (u^*)^T - \frac{\partial J}{\partial u_{\text{in}}} \left( \frac{\partial^2 J}{\partial u^2} \right)_{u=u^*}^{-1},
\]

which is a step of a Newton-Raphson procedure to iteratively approach the optimum. More compactly this can be written as

\[
u^{i+1} = \nu^i - \left( \left( \frac{\partial J}{\partial u} \right)^T \right)^{-1} \nabla J^i,
\]

where \( \nabla J \triangleq (\partial J/\partial u)^T \) is the gradient (i.e. the transposed derivative), \( H \triangleq \partial^2 J/\partial u^2 \) is the Hessian and superscript \( i \) is the iteration counter. Note that the second derivatives in \( H \) provide information about the curvature of \( J \) in point \( u \). The product \(-H^{-1} \nabla J\) is now the search direction, while the stepsize is equal to one. Note that in the special case that \( J(u) \) is a quadratic function of \( u \), equation (1.111) is exact and the single Newton-Raphson step (1.112) leads directly to the optimum. However, normally this is not the case and an iterative approach is required until the change in the objective function becomes smaller than a predefined convergence tolerance. As usual, in a numerical implementation the Hessian is not inverted but the Newton-Raphson step is computed through solving a linear system of equations according to

\[
H^i r^i = -\nabla J^i,
\]

\[
u^{i+1} = \nu^i + \gamma r^i,
\]

where the residual \( r^i \triangleq u^{i+1} - u^i \) can be seen to be identical to the search direction. The scalar \( \gamma \) represents the step size, which has been taken equal to one until now, but which may be chosen differently as will be discussed below. It can be shown that in the neighborhood of the optimum the Newton-Raphson scheme converges quadratically, provided \( J \) is sufficiently smooth. In fact, if the objective function would be exactly quadratic, convergence would be obtained in a single step. Away from the optimum, convergence may be slower while in the presence of inflection points or discontinuities the procedure may diverge or even fail completely. E.g., when \( H \) is singular, no solution to equation (1.114) exists. Moreover, if \( J \) is nonconvex (i.e. if \( H \) is not positive definite for all values of \( u \)), the algorithm will converge to a local optimum which may or may not be equal to the global one.

**Steepest ascent**

Apart from these potential problems, the numerical computation of the Hessian is usually not feasible. Therefore, instead of the true Hessian, an approximation is normally used. In the simplest case the Hessian is replaced by a negative unit matrix which then leads to

\[
u^{i+1} = \nu^i + \gamma' \nabla J^i.
\]

The search direction is now equal to the gradient \( \nabla J^i \) at the current iterate and the algorithm is therefore known as the steepest ascent method (for maximization problems) or steepest descent method (for minimization problems).
Quasi-Newton methods

So-called quasi-Newton methods do not calculate the Hessian explicitly but build up an approximate Hessian with curvature information as the iterations proceed, i.e. they use the information obtained from the successive evaluations of $J$ and the gradient to approximate the curvature in a given direction. Several varieties of these quasi-Newton methods exist. Well known is the limited-memory Broyden Fletcher Goldfarb Shanno (LBFGS) method, where the adjective ‘limited-memory’ indicates that the algorithm does not require matrix computations but only involves matrix-vector products which makes it particularly suited for very large problems. However, in the presence of multiple inequality constraints, which may be alternatingly active and inactive during the iterative optimization procedure, quasi-Newton methods are not always effective, in which case the use of a simple steepest ascent method is sometimes the best option.

1.5.3 Step size

Most iterative gradient-based optimization algorithms require at each iteration step $i$: 1) computation of the search direction $g^i$, and 2) computation of the step size $\gamma$ in that direction:

$$u^{i+1} = u^i + \gamma^i \times g^i,$$

(1.117)

where $g^i$ is in the simplest case just the gradient $\nabla J^i$. Finding an optimal step size is known as a line search procedure and several line search varieties are in use. The more complex ones require multiple function evaluations at points along the search direction. E.g., one can define a parabola through the current iterate and two more points at small step sizes away from the current iterate (along the search direction) and then analytically solve for the step size that maximizes the increase (or decrease) in the objective function value. Another approach is back tracking, in which we simply take a large step in the desired (increasing or decreasing) direction and accept the result if it is higher (or lower) than the current objective function value. However if the result is opposite to what is desired, we have apparently ‘over shot’ the maximum (or minimum) along the search direction and we reduce the step size with a predefined factor and re-evaluate the objective function value. This back tracking is repeated until an increase (or decrease) in objective function value is obtained or until a maximum allowed number of back tracking steps is reached.

Yet other gradient-based numerical optimization methods use a trust region approach in which case the step size is pre-set, but a search direction is determined that maximizes (or minimizes) the objective function value within the corresponding hyper-sphere with radius $\gamma$ in the search space $u$. We will not discuss line-search or trust region methods any further and refer to the text books listed above for detailed information.

1.6 References for Chapter 1


2 Adjoint-based flooding optimization

2.1 Introduction

In this chapter we will consider methods to optimize the management of a reservoir, and in particular the optimization of injection and production rates of wells during water flooding. We restrict ourselves to open-loop optimization, i.e. to optimization during the design phase; see Figure 2.1. We will mainly consider optimization for a given configuration of wells, and we will only briefly discuss optimization in a free configuration, e.g. determining the optimal position of sidetracks or infill wells. A lot of production optimization efforts in the E&P industry are focused on short time scales. Instead we focus on life-cycle optimization, i.e. on optimization over the entire producing life of the reservoir with the aim to optimize ultimate recovery or present value. In particular we will consider gradient-based optimization where the derivative information is obtained through the use of an adjoint equation. (Other names used in the literature for life-cycle optimization are flooding optimization, production optimization or well control optimization). Adjoint-based optimization techniques were introduced in reservoir engineering during the 1970s for computer-assisted history matching; see e.g. the textbook of Oliver et al. (2008) for an overview. About a decade later they also started to be used for the optimization of tertiary recovery processes such as surfactant, polymer, CO₂ or steam flooding; see e.g. Ramirez et al. (1984), Fathi and Ramirez (1984, 1986, 1987), Ramirez (1987), Mehos and Ramirez (1989), Liu et al. (1993), and Liu and Ramirez (1994). The first paper on gradient-based control of water flooding is the one by Asheim (1988), followed by, among others, Virnovsky (1991), Zakirov et al. (1996) and Sudaryanto and Yortsos (2000, 2001). However, industry uptake of these methods was almost absent until the advent of smart well and smart field technology which caused a revival of interest; see Brouwer and Jansen (2004). Since that time a series of publications have appeared covering various aspects of adjoint-based optimization of reservoir flooding while several large reservoir simulation packages have been equipped with the adjoint functionality.

![Figure 2.1: Open-loop optimization.](image-url)
2.2 Problem statement

2.2.1 System model

We consider a given configuration of wells, and we do not address the problem of optimizing the well locations. As indicated in Figure 2.1, the optimization is based on one or more system models. We further restrict our attention to numerical reservoir models in the form of partial differential equations that have been discretized in both space and time. We do not address models for wells and facilities, although they could also be included in the optimization procedure. The optimization variables, indicated as controllable input in Figure 2.1, are typically water injection rates \( q_w \), gas injection rates \( q_g \), or total (i.e. combined gas, oil, and water) production rates \( q \) in the producers. Alternatively one may choose flowing tubing head pressures \( p_{tf} \), or flowing bottom hole pressures \( p_{wq} \) in the injectors and/or the producers, or valve settings \( \alpha \), with elements \( 0 \leq \alpha_i \leq 1 \) representing the dimensionless opening of e.g. wellhead chokes or downhole inflow control valves (ICVs) in a smart well completion. We will start our analysis from an implicitly time-discretized version of the system equations, written in residual form:

\[
g_k(u_k, x_{k-1}, x_k) = 0, \quad k = 1, 2, \ldots, K, \tag{2.1}
\]

where \( g \) is a vector-valued nonlinear function, \( x \) is a vector of state variables, the subscript \( k \) indicates discrete time, and \( K \) is the total number of simulation time steps. The state variables are typically pressures and phase saturations or component accumulations in each grid block. Initial conditions need to be specified in the entire domain, and can be expressed as

\[
x_0 = \bar{x}_0. \tag{2.2}
\]

In addition we consider the output equations:

\[
j_k(u_k, x_k, y_k) = 0, \tag{2.3}
\]

where \( j \) is a vector-valued nonlinear function and \( y \) is a vector of measured output variables which are typically flowing bottom hole pressures \( p_{wq} \) in those wells where the flow rates have been prescribed or phase rates \( \bar{q}_g, \bar{q}_o \) and \( \bar{q}_w \) in those wells where the bottom hole pressures formed the input. In addition \( y \) may contain the interpreted results of field-wide measurements like time-lapse seismic or gravity surveys. For further information on the physics and mathematics behind equations (2.1) and (2.3), see, e.g., Aziz and Settari (1970), Chen et al. (2006) or Chen (2007). For a description of reservoir flow in a systems-and-control format, see Jansen (2013).

---

8 Also referred to as manipulated variables or input variables. The vector \( u \) is referred to as the input vector, control vector or decision vector.

9 Often the discrete-time system equations are represented in explicit form as \( x_k = f_k(u_k, x_{k-1}) \) which can simply be converted to the more general implicit form (also known as residual form) as \( g_k(u_k, x_{k-1}, x_k) \). Similarly, the output equations are often represented explicitly as \( y_k = h_k(u_k, x_k) \). This is related to the implicit form as \( j_k(u_k, x_k, y_k) = y_k - h_k(u_k, x_k) \).

10 The vector-valued functions \( f \) and \( g \) compactly represent the algorithmic operations that take place in an entire reservoir simulator, while \( h \) and \( j \) can represent the operations in a complex measurement system such as the interpretation of time-lapse seismic. These functions are therefore often referred to as nonlinear operators.
2.2.2 Objective function

As in any optimization problem, we aim at maximizing or minimizing an objective. For example, the objective could be to maximize a simple net present value (NPV) of the waterflooding process. Generally, the objective function $\mathcal{J}$ can be expressed as:

$$\mathcal{J}(\boldsymbol{u}_k, \boldsymbol{y}_k (\boldsymbol{u}_k)) = \sum_{k=1}^{K} \mathcal{J}_k (\boldsymbol{u}_k, \boldsymbol{y}_k),$$  \hspace{1cm} (2.4)

where $\mathcal{J}_k$ represents the contribution to $\mathcal{J}$ in each time step $k$ (e.g., oil revenues and water injection and production costs during that time interval, where the costs have a negative value). Note that actually all inputs up to time $k$ may play a role in $\mathcal{J}_k$ in equation (2.4) as can be seen through recursive application of equations (2.1) and (2.3). We should therefore formally write $\mathcal{J}_k(\boldsymbol{u}_k, \boldsymbol{y}_k (\boldsymbol{u}_k, \boldsymbol{x}_k(\boldsymbol{u}_k)))$, but to keep the notation tractable we use $\mathcal{J}_k(\boldsymbol{u}_k, \boldsymbol{y}_k)$ instead. A typical objective function for flooding optimization is given by

$$\mathcal{J}(\boldsymbol{u}_k, \boldsymbol{y}_k (\boldsymbol{u}_k)) = \sum_{k=1}^{K} \left[ \sum_{i=1}^{N_{ij}} \frac{r_{wi} \times (u_{wi,j})_k + \sum_{j=1}^{N_{prod}} (r_{wp} \times |y_{wp,j}|_k + r_o \times |y_{o,j}|_k)}{(1 + b)^{\frac{t_k}{\tau}}} \right], \hspace{1cm} (2.5)$$

where $u_{wi,i}$ is an input variable representing the water injection rate of well $i$ (positive in our sign convention), $y_{wp,j}$ is an output representing the water production rate of well $j$ (negative), $y_{o,j}$ is also an output, representing the oil production rate of well $j$ (also negative), $r_{wi}$ and $r_{wp}$ are the (constant) water injection and production costs (negative, with units $$/m^3$), $r_o$ is the (constant) oil price (positive, $$/m^3$), $\Delta t_k$ is the time interval of time step $k$ in days, $b$ is the discount rate for a reference time interval $\tau$ (which is usually taken as a year), and $N_{ij}$ and $N_{prod}$ are the number of injection wells and production wells respectively. Equation (2.5) can be interpreted as a simplified NPV, i.e., the cumulative discounted cash flow over the producing life of a reservoir, disregarding the capital expenditure for wells, facilities etc. (which are assumed to be fixed). In this example the water injection rates are taken as inputs, and the oil and water production rates as outputs. In a more general case the injection and production rates could either be inputs or outputs, or even change role over time.

2.2.3 Constraints

In practice, the elements of the input vector $\boldsymbol{u}_k$ are often constrained to stay within certain limits. For example, bottom hole pressures in injectors are usually limited to a certain maximum because of the risk of fracturing the rock around the well. Similarly bottom hole pressures in the producers are usually limited to a certain minimum because otherwise it would not be possible to lift the produced fluids to surface. Another form of constraint is when we require the volume of injected water to be equal to the total volume of the fluids produced, a situation known as voidage replacement. Also well rates are usually constrained

---

* Often the objective function $\mathcal{J}$ is split in two parts: one representing the contribution to $\mathcal{J}$ at terminal time $K$, and one representing the contribution before this time. Such a split is relevant for the continuous-time formulation, but for a discrete-time formulation, as used in our text, it offers no added value except to illustrate the analogy with the continuous case.

† Flow into the reservoir is taken as positive, while flow out of the reservoir is negative.
to maximum values. These limitations may all be expressed as equality or inequality constraints, which can be represented in a general form as

$$c_k(u_k, y_k) = 0,$$  \hspace{1cm} (2.6)

and

$$d_k(u_k, y_k) \leq 0.$$  \hspace{1cm} (2.7)

The control problem can now be formulated as

$$\max_{u_{i,k}} J(u_{i,k}, y_{i,k}(u_{i,k})),$$  \hspace{1cm} (2.8)

subject to

- system equations (2.1): \( g_k(u_k, x_{k-1}, x_k) = 0, \quad k = 1, 2, \ldots, K, \)
- initial conditions (2.2): \( x_0 = x_0, \)
- output equations (2.3): \( j_k(u_k, x_k, y_k) = 0, \)
- equality constraints (2.6): \( c_k(u_k, y_k) = 0, \)
- inequality constraints (2.7): \( d_k(u_k, y_k) \leq 0. \)

In the following we will consider problems with

- \( n \) inputs, i.e. \( u \in U \subset \mathbb{R}^n \),
- \( m \) states, i.e. \( x \in X \subset \mathbb{R}^m \),
- \( p \) outputs, i.e. \( y \in Y \subset \mathbb{R}^p \),
- \( q \) equality constraints, i.e. \( c \in C \subset \mathbb{R}^q \),
- \( r \) inequality constraints, i.e. \( d \in D \subset \mathbb{R}^r \).

Note that the sets \( C, D, U, X \) and \( Y \) are subsets of the set of real numbers because their elements are constrained to stay within physical limits, e.g. pressures are always positive and saturations, by definition, have values between zero and one.

### 2.3 Optimal control theory

#### 2.3.1 Adjoint equation - derivation

**Derivatives**

Starting from the optimization problem (2.8) we aim to compute the optimal control \( u_{i,k} \), with the aid of a gradient-based algorithm, which requires the derivatives of \( J(u_{i,k}, y_{i,k}(u_{i,k})) \) with respect to \( u_{i,k} \). An efficient way to compute these derivatives is through the use of an optimization technique known as optimal control; see e.g. Bryson and Ho (1975) or Stengel (1986). The problem in determining the derivatives is the indirect dependence of the variation \( \delta J \) in the objective function on a variation \( \delta u_{i,k} \) of the input. Here, \( \delta u_{i,k} \) means the variation of element \( i \) of vector \( u_k \) at time \( k \). A variation \( \delta u_{i,k} \), at an arbitrary time \( k \), does not only directly influence \( J \) at time \( k \), but also, as follows from recursive application of equation (2.1), the states \( x_{i,k} \), which in turn, through equation (2.3), influence the outputs \( y_{i,k} \) and thus \( J \) at later times. The effect of a single variation \( \delta u_{i,k} \) should therefore be computed, using the chain rule for differentiation, as

$$\frac{dJ}{du_{i,k}} = \left[ \frac{\partial J}{\partial u_{i,k}} + \sum_{j-k} \frac{\partial J}{\partial y_{j,k}} \left( \frac{\partial y_{j,k}}{\partial u_{i,k}} + \frac{\partial y_{j,k}}{\partial x_{j,k}} \frac{\partial x_{j,k}}{\partial u_{i,k}} \right) \right] \frac{\partial u_{i,k}}{\partial u_{i,k}},$$  \hspace{1cm} (2.9)
where \( j \) is a dummy variable within the summation, and where we used the ordinary differential \( d \) instead of the variational symbol \( \delta \) to adhere to the usual notation in the literature. Output equation (2.3) is often an explicit linear algebraic equation such that the terms \( \delta y_j/\delta u_k \) and \( \delta y_j/\delta x_j \) in equation (2.9) can be computed directly\(^\dagger\). Furthermore, the terms \( \delta J/\delta u_k \) and \( \delta J/\delta y_j \) can usually also be computed without problems. However, the term \( \delta x_j/\delta u_k \) causes difficulties because we need to solve the recursive system of discrete-time differential equations (2.1) to connect the state vectors \( x_j, j = k, k + 1, \ldots, K \) to the input \( u_k \) at time \( j = k \). Often the need to determine the derivatives with respect to the individual elements of the input vector \( u_k \) is assumed to be understood tacitly in which case equation (2.9) could be written as

\[
\frac{dJ}{du_k} = \frac{\partial J_k}{\partial u_k} + \sum_{j=k}^{K} \frac{\partial J_j}{\partial y_j} \left( \frac{\partial y_j}{\partial u_k} + \frac{\partial y_j}{\partial x_j} \frac{\partial x_j}{\partial u_k} \right).
\]  

(2.10)

In the following sections we will describe an efficient numerical technique to compute the vector of total derivatives \( dJ/du_k \) with the aid of a so-called adjoint equation.

**Lagrange multipliers**

The complex temporal dependence of the elements in \( \partial x_j/\partial u_k \) can be taken into account by considering equation (2.1) as a set of additional constraints to the optimization problem, and applying the technique of Lagrange multipliers to solve the constrained optimization problem. Moreover, we may formally also consider the initial condition (2.2) and the output equation (2.3) as constraints, and, setting aside the ‘ordinary constraints’ \( c \) and \( d \), we can therefore define a modified objective function

\[
\mathcal{J}(u_{1:k}, x_{0:k}, y_{1:k}, \lambda_{0:k}, \mu_{1:k}) \equiv \sum_{k=1}^{K} \left[ J_k(u, y_k) + \lambda_k^T(x_0 - x_0)\delta_{k-1} + \mu_k^T g_k(u_k, x_{k-1}, x_k) + \mu_k^T j_k(u_k, x_k, y_k) \right],
\]  

(2.11)

where the ‘initial condition constraint’ \( x_0 - x_0 = 0 \), the ‘system constraints’ \( g_{1:k} = 0 \), and the ‘output constraints’ \( j_{1:k} = 0 \) have been ‘adjoined’ to \( J_k \) with the aid of vectors of Lagrange multipliers \( \lambda_0 \), \( \lambda_{1:k} \) and \( \mu_{1:k} \) respectively. The Kronecker delta\(^\ddagger\)\( \delta_{k-1} \) in equation (2.11) ensures that the initial condition constraint is included in the summation.

**Euler-Lagrange equations**

A necessary condition for a maximum of the modified objective function (2.11) is stationarity of the first variation of \( \mathcal{J} \) with respect to all dependent variables. In other words, all first-order derivatives should be equal to zero which, after some reorganization of terms, leads to the following set of equations:

\[
\frac{\partial \mathcal{J}}{\partial u_k} = \frac{\partial J_k}{\partial u_k} + \lambda_k^T \frac{\partial g_k}{\partial u_k} + \mu_k^T \frac{\partial j_k}{\partial u_k} = 0^T, \quad k = 1, 2, \ldots, K,
\]  

(2.12)

\(^\dagger\) All entries of \( \delta y_j/\delta u_k \) for \( j \neq k \) will be equal to zero.

\(^\ddagger\) The Kronecker delta is defined as \( \delta_k = 1 \) if \( k = 0 \), \( \delta_k = 0 \) if \( k \neq 0 \), and can be interpreted as the discrete version of the Dirac delta function. Note that it is indicated with the same symbol as a variation.
\[
\begin{align*}
\frac{\partial J}{\partial x_0} &= \lambda_0^T \frac{\partial g_k}{\partial x_0} + \lambda_0^T = 0^T, \\
\frac{\partial J}{\partial x_k} &= \lambda_{k+1}^T \frac{\partial g_k}{\partial x_k} + \lambda_k^T \frac{\partial g_k}{\partial x_k} + \mu_k^T \frac{\partial j_k}{\partial x_k} = 0^T, \\
\frac{\partial J}{\partial y_k} &= \frac{\partial J}{\partial y_k} + \mu_k^T \frac{\partial j_k}{\partial y_k} = 0^T, \\
\frac{\partial J}{\partial \lambda_0} &= (x_0 - x_0)^T = 0^T, \\
\frac{\partial J}{\partial \mu_k} &= j_k^T (u_k, x_{k-1}, x_k) = 0^T.
\end{align*}
\]

Equations (2.12) to (2.19) are the first-order necessary conditions for an optimum, which are in optimal control theory often called the Euler-Lagrange equations\(^\dagger\). We will discuss their meaning, going from the bottom to the top. The last three equations are identical to output equation (2.3), system equation (2.1) and initial condition (2.2), and are therefore automatically satisfied. Equation (2.16) allows us to compute the Lagrange multipliers \(\mu_{1,K}\).

Next we can use equation (2.15) to compute multiplier \(\lambda_K\) for the final discrete time \(K\), and thereafter equation (2.14) to recursively compute the multipliers \(\lambda_k\) for \(k = K-1, K-2, \ldots, 1\), i.e. backward in time. This last step becomes more clear by rewriting equation (2.14) as

\[
\left( \frac{\partial g_k}{\partial x_k} \right)^T \lambda_k = -\left( \frac{\partial g_{k+1}}{\partial x_k} \right)^T \lambda_{k+1} - \left( \frac{\partial j_k}{\partial x_k} \right)^T \mu_k,
\]

which is a discrete-time differential equation for \(\lambda_k\) that runs backward in time starting from ‘final condition’ \(\lambda_K\). Formally we can solve it as

\[
\lambda_k = -\left( \frac{\partial g_k}{\partial x_k} \right)^T \left[ \left( \frac{\partial g_{k+1}}{\partial x_k} \right)^T \lambda_{k+1} + \left( \frac{\partial j_k}{\partial x_k} \right)^T \mu_k \right]^{-1},
\]

although in practice we will, as usual, solve the system of equations (2.20) for the unknown \(\lambda_k\), rather than explicitly computing the inverse. Equation (2.13) then allows us to compute \(\lambda_0\)\(^\ddagger\). Finally, equation (2.12) represents the effect of changing the control \(u_k\) on the

\(^\dagger\) Usually the term Euler-Lagrange equations is used for a subset of these equations, which implies that equations (2.12) to (2.19) could be referred to as extended Euler-Lagrange equations.

\(^\ddagger\) Formally, equation (2.13) represents the effect of changing the initial condition \(x_0\) on the value of the objective function, while keeping all other variables fixed. However, because we prescribed the initial condition through equation (2.2) this term is in our case only of theoretical relevance and we do not need to compute \(\lambda_0\).
value of the objective function, while keeping all other variables fixed. Because \( \frac{\partial \mathcal{J}}{\partial u_i} = \frac{d \mathcal{J}}{d u_i} \), this is just the expression we were looking for, i.e. equation (2.10), but now with implicitly evaluated derivatives. For a non-optimal control this term is not equal to zero, but then its residual is just the modified gradient that we require to iteratively obtain the optimal control using a gradient-based algorithm.

**Gradient computation**

Computation of the gradient vectors as part of an iterative gradient-based optimization procedure can now be performed according to the following algorithm:

**Algorithm 2.1**

1. Choose an initial control vector \( u_{i,K} \).
2. Compute the states \( x_{i,K} \) and outputs \( y_{i,K} \) using equations (2.1) and (2.3), starting from initial conditions (2.2).
3. Compute the value of the objective function \( \mathcal{J} \) using equation (2.4). If converged stop, else continue.
4. Compute the Lagrange multipliers \( \mu_{i,K} \) and \( \lambda_{i,K} \) using equations (2.16), (2.15) and (2.14).
5. Compute the total derivatives (transposed gradients) \( \frac{d \mathcal{J}}{d u_i} \) of the objective function to the controls from the residuals of equation (2.12) according to:

\[
\frac{d \mathcal{J}}{d u_i} \equiv \frac{\partial \mathcal{J}}{\partial u_i} = \frac{\partial \mathcal{J}}{\partial u_i} + \lambda_{i}^{T} \frac{\partial g_{i}}{\partial u_i} + \mu_{i}^{T} \frac{\partial j_{i}}{\partial u_i}, \quad k = 1, 2, \ldots, K.
\]

(2.22)

Compute an improved estimate of the control vector \( u_{i,K} \), using the derivatives as obtained from equation (2.22), and a gradient-based minimization routine of choice.

6. Return to 2.

Because of its computational efficiency in calculating the gradients of the objective function \( \mathcal{J} \) with respect to the control variables \( u_{i,K} \) the use of optimal control theory is particularly useful in optimization problems with a large number of input variables. Implementation in a numerical reservoir simulator is conceptually relatively simple if the simulator is fully implicit, because in that case the partial derivatives \( \frac{\partial g_{i}}{\partial x_{i}} \) and \( \frac{\partial g_{i+1}}{\partial x_{i}} \), as required in equation (2.20), are already available; see Sarma et al. (2005).

**2.3.2 Lagrangian and Hamiltonian**

Sometimes, in the derivation of the adjoint equations use is made of an auxiliary variable, the *Lagrangian*, defined as

\[
\mathcal{L}_{k} (u_{i}, x_{i-1}, x_{i}, y_{i}, \lambda_{i}, \lambda_{i}, \mu_{i}) \]

\[
\mathcal{J}_{k} (u_{i}, y_{i}) + \lambda_{0}^{T} (x_{0} - \bar{x}_{0}) + \lambda_{i}^{T} g_{i} (u_{i}, x_{i-1}, x_{i}) + \mu_{i}^{T} j_{i} (u_{i}, x_{i}, y_{i}),
\]

(2.23)

with which we can rewrite equation (2.11) as

\[
\mathcal{J} (u_{1,K}, x_{0,K}, y_{1,K}, \lambda_{0,K}, \mu_{1,K}) = \sum_{k=1}^{K} \mathcal{L}_{k} (u_{k}, x_{k-1}, x_{k}, y_{k}, \lambda_{k}, \lambda_{k}, \mu_{k}).
\]

(2.24)

Alternatively, we could consider \( x_{0} \) as an additional control variable, but normally we will not be able to influence its value.
Taking the first variation of equation (2.24) we obtain:

\[ \delta J = \sum_{k=1}^{K} \frac{\partial L}{\partial u_k} \delta u_k + \sum_{k=1}^{K} \frac{\partial L}{\partial x_k} \delta x_{k-1} + \sum_{k=1}^{K} \frac{\partial L}{\partial y_k} \delta y_k + \sum_{k=1}^{K} \frac{\partial L}{\partial \lambda_k} \delta \lambda_k, \quad \delta J = \sum_{k=1}^{K} \frac{\partial L}{\partial u_k} \delta u_k + \sum_{k=1}^{K} \frac{\partial L}{\partial x_k} \delta x_{k-1} + \sum_{k=1}^{K} \frac{\partial L}{\partial y_k} \delta y_k + \sum_{k=1}^{K} \frac{\partial L}{\partial \lambda_k} \delta \lambda_k. \tag{2.25} \]

By splitting off the terms for \( k = 0 \) and \( k = K \) in the second and third terms at the right-hand side of equation (2.25) respectively and reordering the results we obtain

\[ \delta J = \sum_{k=1}^{K} \frac{\partial L}{\partial u_k} \delta u_k + \sum_{k=1}^{K} \frac{\partial L}{\partial x_k} \delta x_{k-1} + \sum_{k=1}^{K} \frac{\partial L}{\partial y_k} \delta y_k + \sum_{k=1}^{K} \frac{\partial L}{\partial \lambda_k} \delta \lambda_k. \tag{2.26} \]

Requiring stationarity of \( \delta J \) for all variations, and using the definition of the Lagrangian (2.23) to work out the terms in equation (2.26), we recover the Euler–Lagrange equations (2.12) to (2.19). Sometimes another auxiliary function, the Hamiltonian, is used. Both the Lagrangian and the Hamiltonian have their origin in classical mechanics; see e.g. Landau and Lifshitz (1960). Use of the Hamiltonian is more appropriate when the system equations are expressed in explicit form \( x_i = f_i(u_k, x_{k-1}) \), in which case we have to adjoin the expression \( x_k - f_i(u_k, x_{k-1}) \) to the objective function. The Hamiltonian is then defined as

\[ \mathcal{H}_k \triangleq J_k(u_k, x_k) + \lambda_0^T (x_0 - \bar{x}_k) \delta x_{k-1} - 2 \lambda_k^T f_k(u_k, x_{k-1}) + \mu_k^T j_k(u_k, x_k, y_k), \tag{2.27} \]

such that we can rewrite equation (2.11) as

\[ \overline{J}(u_{0:k}, x_{0:k}, y_{1:k}, \lambda_{0:k}, 0_k:k) = \sum_{k=1}^{K} [x_k + \mathcal{H}_k(u_k, x_{k-1}, y_k, \lambda_0, \lambda_k, \mu_k)]. \tag{2.28} \]

### 2.3.3 Adjoint equation — interpretations*

**Tangent linear model**

The adjoint formulation provides a computationally efficient means of computing derivatives \( dJ/du_k \) by considering the propagation of perturbations \( \delta u_k \) (or \( \delta u_k \)) through the system with the aid of a transposed tangent linear system model. This can be emphasized by rewriting the Jacobians used in the adjoint formulation as:

\[ \overline{E}_k \triangleq \frac{\partial g_k}{\partial x_k}, \quad \overline{A}_k \triangleq \frac{\partial g_k}{\partial x_{k-1}}, \quad \overline{B}_k \triangleq \frac{\partial g_k}{\partial u_k}, \]  

\[ \overline{C}_k \triangleq \frac{\partial j_k}{\partial x_k}, \quad \overline{D}_k \triangleq \frac{\partial j_k}{\partial u_k}, \quad \overline{F}_k \triangleq \frac{\partial j_k}{\partial y_k}, \quad \overline{G}_k \triangleq \frac{\partial j_k}{\partial \lambda_k}. \tag{2.29, 2.30, 2.31} \]

\[ \overline{C}_k \triangleq \frac{\partial j_k}{\partial x_k}, \quad \overline{D}_k \triangleq \frac{\partial j_k}{\partial u_k}, \quad \overline{F}_k \triangleq \frac{\partial j_k}{\partial y_k}. \tag{2.32, 2.33, 2.34} \]

where the overbars indicate tangent matrices and the hats indicate that we use a generalized state space formulation, in line with the notation convention of Jansen (2013). To simplify the notation we will drop the overbars in the remainder of this chapter. With the aid of

---

* C.f. equations (2.30) to (2.35) in Jansen (2013). Note that here we use derivatives with respect to the time-discretized variables \( u_k, x_{k-1}, x_k \) and \( y_k \), whereas in equations (2.30) to (2.35) the derivatives were taken with respect to the time-continuous variables \( u(t), x(t), x(t) \) and \( y(t) \).
expressions (2.29) to (2.34) we can rewrite equations (2.12) to (2.16), and after reorganizing the terms we obtain the following computational sequence:

\[
\mathbf{F}_k^T \lambda_k = - \left( \frac{\partial J_k}{\partial y_k} \right)^T, \quad k = 1, 2, \ldots, K, \quad (2.35)
\]

\[
\mathbf{E}_k^T \lambda_K = - \mathbf{C}_k^T \mu_k, \quad (2.36)
\]

\[
\mathbf{E}_k^T \lambda_k = - \mathbf{A}_k^T \lambda_{k+1} - \mathbf{C}_k^T \mu_k, \quad k = 1, 2, \ldots, K-1, \quad (2.37)
\]

\[
\lambda_0 = - \mathbf{A}_1^T \lambda_1,
\]

\[
\left( \frac{\partial J}{\partial u_k} \right)^T = \left( \frac{\partial J_k}{\partial u_k} \right)^T + \mathbf{B}_k^T \lambda_k + \mathbf{D}_k^T \mu_k, \quad k = 1, 2, \ldots, K. \quad (2.39)
\]

The linear discrete-time ‘backward’ differential equation (2.37), which describes the evolution of the Lagrange multipliers, can be interpreted as the dual of a tangent-linear ‘forward’ equation

\[
\mathbf{E}_k \mathbf{x}_k = \hat{\mathbf{A}}_{k-1} \mathbf{x}_{k-1} + \hat{\mathbf{B}}_k \mathbf{u}_k, \quad (2.40)
\]

which can be obtained by linearization of the nonlinear discrete-time system equation \( g_k (\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k) = 0 \) and which describes the tangent evolution of the states. Note that in our derivation of the adjoint equations we first discretized and then linearized the system equation. It is also possible to use the reversed order and derive the adjoint equations from the continuous-time system equation and thereafter discretize them in time. The two different approaches may lead to slightly different formulations. It is important that the Jacobians used in the forward and backward equations are identical. This implies that they should both be obtained through differentiation with respect to either time-discretized or time-continuous variables, to ensure that the equations are truly each others adjoint.

**Super vectors**

An alternative way to interpret the adjoint equation has been presented in the reservoir engineering literature by Rodrigues (2006) and Kraaijevanger et al. (2007). Instead of considering the individual vectors \( \mathbf{u}_k, \mathbf{x}_k, \) and \( \mathbf{y}_k \) for time steps \( k = 1, 2, \ldots, K \) these authors combined them in super vectors or concatenated vectors defined as

\[
\mathbf{u} \triangleq \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_K \end{bmatrix}, \quad \mathbf{x} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}, \quad \mathbf{y} \triangleq \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_K \end{bmatrix}, \quad (2.41, 2.42, 2.43)
\]

with which we can write the system equations (2.1) and output equations (2.3) in super vector form as

\[
g(\mathbf{u}, \mathbf{x}) \triangleq \begin{bmatrix} g_1 (\mathbf{u}_1, \mathbf{x}_0, \mathbf{x}_1) \\ g_2 (\mathbf{u}_2, \mathbf{x}_1, \mathbf{x}_2) \\ \vdots \\ g_K (\mathbf{u}_K, \mathbf{x}_{K-1}, \mathbf{x}_K) \end{bmatrix}, \quad j(\mathbf{u}, \mathbf{x}, \mathbf{y}) \triangleq \begin{bmatrix} j_1 (\mathbf{u}_1, \mathbf{x}_1, \mathbf{y}_1) \\ j_2 (\mathbf{u}_2, \mathbf{x}_2, \mathbf{y}_2) \\ \vdots \\ j_K (\mathbf{u}_K, \mathbf{x}_K, \mathbf{y}_K) \end{bmatrix}. \quad (2.44, 2.45)
Also equation (2.10) can now be written more simply as
\[
\frac{d\mathcal{J}}{du} = \frac{\partial \mathcal{J}}{\partial u} + \frac{\partial \mathcal{J}}{\partial y} \left[ \frac{\partial y}{\partial u} + \frac{\partial y}{\partial x} \frac{\partial x}{\partial u} \right].
\] (2.46)

The term \( \partial x / \partial u \) can be computed, in theory, through implicit differentiation as
\[
\frac{\partial x}{\partial u} = -\left( \frac{\partial g}{\partial x} \right)^{-1} \frac{\partial g}{\partial u},
\] (2.47)

and similar expressions can be obtained for the other terms\(^* \). With these, equation (2.46) can be rewritten as
\[
\frac{d\mathcal{J}}{du} = \frac{\partial \mathcal{J}}{\partial u} - \frac{\partial \mathcal{J}}{\partial y} \left( \frac{\partial j}{\partial y} \right)^{-1} \frac{\partial j}{\partial u} - \frac{\partial \mathcal{J}}{\partial y} \left( \frac{\partial j}{\partial y} \right)^{-1} \frac{\partial j}{\partial x} \left( \frac{\partial g}{\partial x} \right)^{-1} \frac{\partial g}{\partial u},
\] (2.48)

where \( M = mK \), \( N = nK \) and \( P = pK \), with \( m, n, p \) and \( K \) representing the number of input variables, state variables, output variables and time steps respectively. We stress that super matrices are not used for actual computations because their storage requirements would be far too large for realistic applications, even although they have a sparse structure. However they do give us insight in the amount of operations involved in computing derivatives using different methods. In particular consider term III in equation (2.48). Evaluation of this term from right to left requires a series of matrix-vector multiplications, whereas evaluation from left to right requires only matrix-vector computations which is computationally much more efficient. If we define the auxiliary super vectors
\[
\mu^T \triangleq -\frac{\partial \mathcal{J}}{\partial y} \left( \frac{\partial j}{\partial y} \right)^{-1} , \quad \lambda^T \triangleq -\frac{\partial \mathcal{J}}{\partial y} \left( \frac{\partial j}{\partial y} \right)^{-1} \frac{\partial j}{\partial x} \left( \frac{\partial g}{\partial x} \right)^{-1},
\] (2.49)

we can rewrite equation (2.48) as
\[
\frac{d\mathcal{J}}{du} = \frac{\partial \mathcal{J}}{\partial u} + \mu^T \frac{\partial j}{\partial u} + \lambda^T \frac{\partial g}{\partial u}.
\] (2.50)

The auxiliary super vectors \( \lambda \) and \( \mu \) can be interpreted as Lagrange multipliers and equation (2.50) is therefore the super-vector form of equation (2.12) and represents the effect of changing the control \( u \) on the value of the objective function \( \mathcal{J} \), while keeping all other variables fixed. From detailed inspection of the structure of the super matrices in equation (2.48) it follows that reverse evaluation of term III also implies reverse operation in time; see Kraaijevanger et al. (2007). This is in line with the ‘backward’ computation of the Lagrange multipliers in equation (2.20). Finally, comparison of equations (2.46) and (2.48) to equations (1.36) and (1.42) respectively illustrates the parallel between the simple constrained optimization example treated in Section 1.1 and the large-scale discrete-time optimal control problem treated here.

\(^*\) To compute \( \partial y / \partial u \) for \( j(u,x,y(u,x)) = 0 \), write \( d\mathcal{J} = (\partial j / \partial u) du + (\partial j / \partial x) dx + (\partial j / \partial y) dy = 0 \) from which follows \( dy / du = -(\partial j / \partial y)^{-1} (\partial j / \partial x)(dx / du) \). Because the function \( j \) does not specify an explicit relationship between \( x \) and \( u \) we have \( dx / du = 0 \) and therefore \( \partial y / \partial u = dy / du = -(\partial j / \partial y)^{-1} (\partial j / \partial u) \). In a similar way we find that \( \partial y / \partial x = dy / dx = -(\partial j / \partial y)^{-1} (\partial j / \partial x) \).
Repeated implicit differentiation*

Yet another interpretation† of the adjoint equations can be obtained by starting from equation (2.10) again:

\[
\frac{dJ}{du_k} = \sum_{j=m}^{k} \frac{\partial J}{\partial y_j} \left( \frac{\partial y_j}{\partial u_k} + \frac{\partial y_j}{\partial x_j} \frac{\partial x_j}{\partial u_k} \right),
\]

(2.51)

where we have left out the term \( \partial J_k / \partial u_k \) to simplify the exposition. As discussed in Section 2.3.1 the term \( \partial x_j / \partial u_k \) is problematic because we need to connect the state vectors \( x_k \) at times \( j = k, k + 1, \ldots, K \) to the input \( u_k \) at time \( j = k \). This can be expressed as

\[
\frac{\partial x_j}{\partial u_k} = \frac{\partial x_j}{\partial x_{j-1}} \frac{\partial x_{j-1}}{\partial x_{j-2}} \cdots \frac{\partial x_{j+1}}{\partial x_j} \frac{\partial x_j}{\partial u_k},
\]

(2.52)

where the terms of the form \( \partial x_{k+1} / \partial x_k \) can, in theory, be computed through implicit differentiation according to‡

\[
\frac{\partial x_k}{\partial x_{k-1}} = -\left( \frac{\partial g_k}{\partial x_k} \right)^{-1} \frac{\partial g_k}{\partial x_{k-1}} = -\left( \hat{E}_k \right)^{-1} \hat{A}_k,
\]

(2.53)

where the last term is expressed in terms of the matrices introduced in equations (2.29) and (2.30). Similarly the terms \( \partial y_j / \partial u_k \) and \( \partial x_j / \partial u_k \) in equations (2.51) and (2.52) can be computed through implicit differentiation using equations (2.1) and (2.3). If we furthermore define the matrices

\[
A_k \triangleq -\left( \hat{E}_k \right)^{-1} \hat{A}_k, \quad B_k \triangleq -\left( \hat{E}_k \right)^{-1} \hat{B}_k,
\]

(2.54, 2.55)

\[
C_k \triangleq -\left( \hat{F}_k \right)^{-1} \hat{C}_k, \quad D_k \triangleq -\left( \hat{F}_k \right)^{-1} \hat{D}_k,
\]

(2.56, 2.57)

equation (2.51) can be rewritten as

\[
\frac{dJ}{du_k} = \sum_{j=m}^{k} \frac{\partial J}{\partial y_j} \left( D_k + C_j A_j A_{j-1} \cdots A_{j+2} A_{j+1} B_k \right),
\]

(2.58)

which illustrates that the total derivative \( dJ / du_k \) can be obtained formally through application of the chain rule and repeated implicit differentiation. Equation (2.58) can also be written, in transposed form, as

\[
\left[ \frac{dJ}{du_k} \right]^T = \sum_{j=m}^{k} \left( B_j^T A_{j+1}^T A_{j+2}^T \cdots A_{j+1}^T A_{j+2}^T C_j^T + D_k^T \right) \left[ \frac{\partial J}{\partial y_j} \right]^T.
\]

(2.59)

Evaluation of the terms in the \( m \times p \) matrix in this equation from left to right, i.e. forward in time, requires a series of matrix-matrix multiplications, whereas evaluation from right to left

---

*Personal communication A.W. Heemink, Delft Institute of Applied Mathematics, TU Delft.
† Here we make use of the fact that the functions \( g_k \) and \( j_k \) can be expressed as \( g_k (u_k, x_{k+1}, x_k (u_k, x_{k-1})) = 0 \) and \( j_k (u_k, x_k, y_k (u_k, x_k)) = 0 \) respectively. See also the footnote on page 50.
i.e. backward in time, requires only matrix-vector computations which is computationally much more efficient. Even although the matrices are typically very sparse, it is computationally much more efficient to perform a large number of matrix-vector computations than the same number of matrix-matrix computations.

Remarks

• The computation of the total derivative $dJ/du_k$ using a forward-in-time computation of the $m \times p$ matrix in equation (2.58) or (2.59) is known as the forward sensitivity method. Similarly, computation of $dJ/du_k$ using a backward-in-time computation is known as the backward sensitivity method (or the adjoint method). Because we consider only one objective function $J$, and a multivariate input vector $u$, the backward method is computationally more efficient. If we would consider a vector of multiple objective functions, the forward method would become the computationally preferred choice once the number of objective functions exceeded the number of input variables†.

• Just as the matrix sequence $A_j A_{j-1} \cdots A_{k+2} A_{k+1}$ in equation (2.58) represents the partial derivatives $(\partial x_j / \partial x_{j-1}) (\partial x_{j-1} / \partial x_{j-2}) \cdots (\partial x_{k+2} / \partial x_{k+1}) (\partial x_{k+1} / \partial x_k)$, see equations (2.51) to (2.58), the matrix sequence $A^T_{k+1} A^T_{k+2} \cdots A^T_{j-1} A^T_{j}$ in equation (2.59) can be interpreted to also represent partial derivatives, but now expressed in terms of Lagrange multipliers, and reversed in time: $(\partial x_k / \partial x_{k+1}) (\partial x_{k+1} / \partial x_{k+2}) \cdots (\partial x_{j-2} / \partial x_{j-1}) (\partial x_{j-1} / \partial x_j)$.

2.4 Constrained optimization

2.4.1 Input constraints and output constraints

Several authors discussed the incorporation of constraints in the optimal control problem for flooding optimization in reservoir engineering. Many of those concern partial and sometimes heuristic solutions, only valid for particular types of constraints, while some others represent more systematic studies, valid for a much broader range of constraint equations. Specific references are given in the remainder of this chapter.

A relatively simple situation occurs when the constraints are specified only in terms of the input variables:

$$c_k(u_k) = 0, \quad (2.60)$$

$$d_k(u_k) \leq 0, \quad (2.61)$$

where $c$ and $d$ are vector-valued equality and inequality constraint functions respectively. An example of inequality constraints (2.61) occurs in a situation where we use water injection rates as control variables and specify maximum values for the individual rates. An example of equality constraints (2.60) is a similar situation with water injection rates as controls, however under the constraint that the sum of the rates remains constant. A more complicated situation occurs when the constraints also depend on the state variables or the outputs. In particular we will consider output constraints† of the form

† There is no need for a large number of objective functions in case of flooding optimization as discussed in this chapter. However, there may be other applications, e.g. computer-assisted history matching, where many objective functions may be required.

† Because the outputs depend directly on the states, output constraints are special case of state constraints. Sometimes these are called state-path constraints to emphasize that they contain dependencies along the trajectory in state-time space. In particular, the state values $x_k$ at time $k$ depend on all inputs $u_{1:k}$.
\[ c_k(u_k, y_k) \equiv c_k(u_k, y_k(u_k, x_k(u_k))) = 0, \quad (2.62) \]

\[ d_k(u_k, y_k) \equiv d_k(u_k, y_k(u_k, x_k(u_k))) \leq 0, \quad (2.63) \]

where \( y_k \) is a function of the input vector \( u_k \) and the state vector \( x_k \), which, in turn, is a function of the input vectors \( u_{1:k} \) as follows from recursive application of equations (2.1) and (2.3). An example of equality constraints (2.62) occurs when we specify a voidage replacement condition by requiring that the total amount of produced fluids equals the total amount of injected fluids at each moment in time. Inequality constraints (2.63) occur e.g. when we specify well constraints in terms of bottom hole pressures, or in terms of total rates, phase rates or phase fractions in production wells. In general, such output constraints are much more difficult to handle than the input constraints (2.60) and (2.61).

### 2.4.2 Bound constraints on the input

A special case of inequality constraints, known as bound constraints, are those that limit the input variables to stay within upper and lower bounds:

\[ u^-_k \leq u_k \leq u^+_k, \quad (2.64) \]

where \( u^- \) and \( u^+ \) are vectors of lower and upper bounds respectively. Wang et al. (2009) addressed this problem with a nonlinear transformation

\[ \tilde{u}_k = \ln \left( \frac{u_k - u^-_k}{u^+_k - u^-_k} \right). \quad (2.65) \]

The elements of the transformed vector \( \tilde{u}_k \) have lower and upper bounds of \(-\infty\) and \(\infty\) respectively such that they can be can be optimized over the entire real axis. Van Essen et al. (2009) applied a different method to cope with bound constraints on the inputs which makes use of a projected gradient

\[ \frac{\partial J}{\partial u_k}_{\text{proj}} = P^\perp \frac{\partial J}{\partial u_k}. \quad (2.66) \]

Here \( P^\perp \) is an orthogonal-projection matrix defined as

\[ P^\perp \equiv I - \left( \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k} \right)^T \left( \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k} \left( \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k} \right)^T \right)^{-1} \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k}, \quad (2.67) \]

where \( \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k} \) is the active-constraint matrix, i.e. a matrix of vectors tangent to the active constraints \( \tilde{d} \); c.f. equation (1.96) in Section 1.4.5. Matrix \( P^\perp \) projects the vector \( \frac{\partial J}{\partial \tilde{u}_k} \) on the null space of \( \frac{\partial \tilde{d}_k}{\partial \tilde{u}_k} \). This gradient projection method is most effective for linear input constraints which could be equality or inequality group constraints or bound constraints on individual inputs. However, the situation becomes more complicated for nonlinear input constraints, and in particular for output constraints, whether linear or nonlinear. We refer to Luenberger and Ye (2010) for an extensive analysis of the method.

### 2.4.3 External constraint handling

The simplest way to cope with other equality and inequality constraints is to use gradient-based optimization methods for constrained optimization as programmed in dedicated
software packages. These external packages then use the reservoir simulator as ‘function evaluator’ and typically require the following information:

1. Values of the constraint functions \( c_k \) and \( d_k \), and their derivative matrices \( [\partial c_k/\partial \mathbf{u}_k] \) and \( [\partial d_k/\partial \mathbf{u}_k] \).

2. Values of the objective function \( J \), and the derivative vector \( dJ/du_k \) as obtained from equation (2.22).

In case of input constraints we can obtain the constraint derivative matrices \( [\partial c_k/\partial \mathbf{u}_k] \) and \( [\partial d_k/\partial \mathbf{u}_k] \) without the need to perform reservoir simulations. However, in case of output constraints the situation is more complicated and determination of the constraint derivative matrices requires the solution of an adjoint equation for each element of \( c_k \) and \( d_k \). In practice this is nearly always a much too large number of adjoint simulations to be computationally feasible, and therefore some form of approximate treatment is required.

2.4.4 Internal constraint handling

Equality constraints

As an alternative to handling the constraints externally, we can incorporate them in the definition of the modified objective function. Restricting the analysis to equality constraints for the moment this results in (Jansen, 2011):

\[
\tilde{J}(\mathbf{u}_{1,K}, \mathbf{x}_{0,K}, \mathbf{y}_{1,K}, \lambda_{0,K}, \lambda_{1,K}, \nu_{1,K}) \triangleq \sum_{k=1}^{K} \left[ J_k (\mathbf{u}_k, y_k) + \lambda_0^T (x_0 - \mathbf{x}_0) \delta_{k-1} + \lambda^T_k \mathbf{g}_k (\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k) + \mathbf{\mu}_k^T \mathbf{j}_k (\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k) + \nu_k^T \mathbf{c}_k (\mathbf{u}_k, \mathbf{y}_k) \right],
\]

which is identical to equation (2.11) except for the addition of the \( q \) equality constraints \( \mathbf{c}_{1,K} = 0 \) which have been adjoined to \( J \) with the aid of \( q \) Lagrange multipliers \( \nu_{1,K} \). Following the same procedure as in Section 2.3.1, we obtain the extended Euler-Lagrange equations

\[
\frac{\partial \tilde{J}}{\partial \mathbf{u}_k} = \frac{\partial J_k}{\partial \mathbf{u}_k} + \lambda_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{u}_k} + \mathbf{\mu}_k^T \frac{\partial \mathbf{j}_k}{\partial \mathbf{u}_k} + \nu_k^T \frac{\partial \mathbf{c}_k}{\partial \mathbf{u}_k} = 0_T, \quad k = 1, 2, \ldots, K,
\]

\[
\frac{\partial \tilde{J}}{\partial \mathbf{x}_0} = \lambda_0^T \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_0} + \lambda_0^T = 0_T,
\]

\[
\frac{\partial \tilde{J}}{\partial \mathbf{x}_k} = \lambda_k^T \frac{\partial \mathbf{g}_{k-1}}{\partial \mathbf{x}_k} + \lambda_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k} + \mathbf{\mu}_k^T \frac{\partial \mathbf{j}_k}{\partial \mathbf{x}_k} = 0_T, \quad k = 1, 2, \ldots, K - 1,
\]

\[
\frac{\partial \tilde{J}}{\partial \mathbf{x}_K} = \lambda_K^T \frac{\partial \mathbf{g}_{K-1}}{\partial \mathbf{x}_K} + \mathbf{\mu}_K^T \frac{\partial \mathbf{j}_K}{\partial \mathbf{x}_K} = 0_T,
\]

\[
\frac{\partial \tilde{J}}{\partial \mathbf{y}_k} = \frac{\partial J_k}{\partial \mathbf{y}_k} + \mathbf{\mu}_k^T \frac{\partial \mathbf{j}_k}{\partial \mathbf{y}_k} + \nu_k^T \frac{\partial \mathbf{c}_k}{\partial \mathbf{y}_k} = 0_T, \quad k = 1, 2, \ldots, K,
\]
\[ \frac{\partial J}{\partial \lambda_0} = (x_0 - \hat{x}_0)^T = 0^T, \quad (2.74) \]

\[ \frac{\partial J}{\partial \lambda_k} = g_k^T (u_k, x_{k-1}, x_k) = 0^T, \quad k = 1, 2, \ldots, K, \quad (2.75) \]

\[ \frac{\partial J}{\partial u_k} = j_k^T (u_k, x_k, y_k) = 0^T, \quad k = 1, 2, \ldots, K. \quad (2.76) \]

\[ \frac{\partial J}{\partial \nu_k} = c_k^T (u_k, y_k) = 0^T, \quad k = 1, 2, \ldots, K. \quad (2.77) \]

In comparison with equations (2.12) and (2.16), equations (2.69) and (2.73) each have an additional term that contains the \( q \) unknown multipliers \( v_k \). At first sight we might expect that the additional set of \( q \) equations (2.77) should enable us to solve for the additional multipliers, but because they do not explicitly contain the multipliers we cannot use them directly and need a somewhat more elaborate procedure. As before we can choose an initial control strategy \( u_{1:K} \) to compute numerical values for the corresponding state and output variables \( x_{1:K} \) and \( y_{1:K} \) and for the Jacobians \( \partial g_k / \partial u_k, \partial g_k / \partial x_{k-1} \), etc. During this forward integration of the system equations we may, or may not, choose to obey some or all of the constraints \( c_{i,K} = 0 \) using some heuristic, i.e. generally non-optimal, strategy. With the aid of equations (2.69), (2.72) and (2.73) we can set up a system of equations for the multipliers \( \lambda_k, \mu_k \) and \( v_k \) at the final discrete time \( K \):

\[
\begin{bmatrix}
\lambda_k^T \\
\mu_k^T \\
v_k^T
\end{bmatrix}_{1 \times (n+p+q)} \begin{bmatrix}
\frac{\partial g_k}{\partial u_k} & \frac{\partial g_k}{\partial x_k} & 0 \\
\frac{\partial j_k}{\partial u_k} & \frac{\partial j_k}{\partial x_k} & \frac{\partial j_k}{\partial y_k} \\
\frac{\partial c_k}{\partial u_k} & 0 & \frac{\partial c_k}{\partial y_k}
\end{bmatrix}_{(n+p+q) \times (m+n+p)} = \begin{bmatrix}
\frac{\partial J_k}{\partial u_k} \\
\frac{\partial J_k}{\partial y_k}
\end{bmatrix}_{1 \times (m+n+p)}.
\]

In the special case that there are \( q = m \) linearly independent constraints\(^\dagger\), i.e. just as many constraints as there are control variables, the system matrix in equation (2.78) is square and invertible and allows us to solve for the vector of Lagrange multipliers \( [\lambda_k^T \mu_k^T v_k^T]^T \). Note that in this special case all elements of \( \partial J_k / \partial u_k \) are identical to zero, i.e. all elements of \( u_k \) are fixed and, at least for this time step, there is nothing left to optimize. Normally, however, there will be fewer constraints than control variables in which case the system matrix in equation (2.78) becomes rectangular. In that case we may decide beforehand which \( q \) elements of \( u_k \) are fixed and which \( m-q \) elements are left free. Indicating the \( q \times 1 \) vector of fixed input variables with \( \hat{u} \) we can then replace the derivative matrices \( \partial J_k / \partial u_k \) in equation (2.78) by reduced-size matrices \( \partial \hat{J} / \partial \hat{u}_k \) such that the system can be solved again. Thereafter we can then substitute the multipliers in equation (2.69) and compute the \( m-q \) non-zero elements of \( \partial J_k / \partial u_k \). Going backwards in time we can compute the non-zero elements of

\(^\dagger\) With ‘linearly independent constraints’ we mean that the Jacobian \( \partial c_k / \partial u_k \) is regular, i.e. that the linearized constraints are linearly independent.
\[ \frac{\partial \mathcal{J}_k}{\partial \mathbf{u}_k}, k = K-1, \ldots, 1 \] in a similar fashion, where the system of equations for the multiplier vector \([\lambda^T, \mu^T, v^T]\) is formed with the aid of equations (2.69), (2.71) and (2.73), i.e. for the case of reduced-size matrices \( \frac{\partial \mathbf{u}}{\partial \lambda} \):

\[
\begin{bmatrix}
\frac{\partial g_k}{\partial \mathbf{u}_k} & \frac{\partial g_k}{\partial \mathbf{x}_k} & 0 \\
\frac{\partial j_k}{\partial \mathbf{u}_k} & \frac{\partial j_k}{\partial \mathbf{x}_k} & \frac{\partial j_k}{\partial \mathbf{y}_k} \\
\frac{\partial c_k}{\partial \mathbf{u}_k} & 0 & \frac{\partial c_k}{\partial \mathbf{y}_k}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial \mathcal{J}_{k+1}}{\partial \mathbf{u}_k} & \lambda^T \frac{\partial g_{k+1}}{\partial \mathbf{x}_k} & \frac{\partial \mathcal{J}_{k+1}}{\partial \mathbf{y}_k}
\end{bmatrix},
\]

\( k = K-1, K-2, \ldots, 1 \).

Thereafter equation (2.69) gives the reduced gradient (c.f. equation (2.22))

\[
\frac{d \mathcal{J}}{d \mathbf{u}_k} = \frac{\partial \mathcal{J}_k}{\partial \mathbf{u}_k} + \lambda^T \frac{\partial g_k}{\partial \mathbf{u}_k} + \mu^T \frac{\partial j_k}{\partial \mathbf{u}_k} + v^T \frac{\partial c_k}{\partial \mathbf{u}_k},
\]

\( k = 1, 2, \ldots, K \),

where \( \mathbf{u} \) indicates the \((m-q) \times 1\) vector of free input variables.

### Inequality constraints

To incorporate state or output inequality constraints in the optimization, various methods originating from nonlinear programming are available, see e.g. Rao (1996) or Luenberger and Ye (2010), and several of them have been applied in water flooding optimization. Virnovski (1991), De Montleau et al. (2006) and Kraaijevanger et al. (2007) have implemented variations of the generalized reduced gradient (GRG) method. The latter two introduced so-called slack variables to transform inequality constraints to equality constraints according to

\[
d_k (\mathbf{u}_k, \mathbf{y}_k) - s_k = 0, \quad s_k \geq 0.
\]

The slack variables \( s_k \) can now be treated as bounded input variables, while the equality constraints (2.81) can be treated using Lagrange multipliers as described in Section 0 above. Note that although the use of slack variables at first sight appears to remove the difficulty of dealing with state and output constraints, this is not really the case because a non-trivial strategy is required to select the ‘fixed’ input variables \( \hat{\mathbf{u}}_k \) and \( \hat{s}_k \) and to switch between active and inactive inequality constraints. This may require many multiple forward and backward (adjoint) simulations before an acceptable search direction has been found. An alternative method to incorporate constraints was proposed by Sarma et al. (2008) which makes use of a constraint-lumping technique in combination with a search strategy known as Zoutendijk’s method to obtain an approximate constrained gradient. Kourounis et al. (2014) also used constraint lumping in combination with a set of constraint handling methods in an external optimizer. Kourounis and Schenk (2015) compared this method to a more formal one without lumping (i.e. by applying all the relevant constraints at every time step) and to a heuristic approach where the constraints are handled by the simulator. They found that the formal approach typically resulted in higher optima, although at the price of reduced computational efficiency. Yet another method was proposed by Chen et al. (2010) who
applied an *augmented Lagrangian* method which uses a penalty function that allows for temporary violation of the constraints during the iterative optimization procedure. Another penalty function approach, but restricted to operate within the feasible region, known as a *barrier method*, was proposed by Suwartadi et al. (2009, 2010). Further possibilities for constrained optimization have been discussed by Sarma et al. (2005, 2008), Dehdari et al. (2012) and Codas et al. (2015) but a systematic comparison of all methods, in particular for realistically sized reservoir models is lacking. Two implementations in proprietary professional reservoir simulators have been described; see De Montleau et al. (2006) and Kraaijevanger et al. (2007). Although it is clear that they both use a form of the GRG method, the details of the strategy to search for active constraints cannot be inferred from these publications.

### 2.5 Simple example

We consider a small-scale example originating from Brouwer (2004) and first published in Brouwer and Jansen (2004), see Figure 2.2. Although later criticized for its unrealistic configuration, it has been used in several publications because it effectively shows the potential for the application of model-based optimization techniques to increase oil recovery. Here we present some unpublished additional results from Brouwer (2004) that further illustrate this potential.

![Figure 2.2: Schematic representation of a horizontal reservoir with two horizontal ‘smart’ wells. After Brouwer and Jansen (2004).](image)

*This example was taken verbatim from Jansen (2011).*
Figure 2.3: Seven snapshots in time. The first and third rows represent water saturation (in black), while the second and fourth rows depict streamline patterns. The crosses indicate closure of the ICVs in the corresponding segments. The larger figure at the bottom right represents the permeability distribution in the reservoir where black and white indicate high and low permeability respectively. After Brouwer (2004).

The schematic two-dimensional, square reservoir depicted in Figure 2.2 is operated by injecting water in a horizontal injection well, while producing oil, and after a while also water, from a horizontal production well. For details about the reservoir and fluid properties of this example we refer to Brouwer and Jansen (2004). Each of the wells is completed with a tubing, and the annular space between the well and the tubing is divided in segments with the aid of rubber packers. The segments are connected to the tubing through ICVs which allow for individual control of the injection or production from the corresponding well segments. Moreover, it is assumed that the pressure and total flow rates in each of the segments can be measured, either directly, or through indirect ‘soft sensing’. Such smart wells are installed increasingly, and typically contain three to five ICVs. Although the example uses an unrealistically large number of ICVs (45 per well), similar results have been obtained with a much smaller number (say five) ICVs per well. Moreover, the concept of model-based subsurface flow control is not restricted to reservoirs operated with smart wells, but may also be applied using conventional, vertical injectors and producers. We note, however, that the scope for control, if present at all, strongly depends on factors like reservoir geometry, and rock and fluid properties. Figure 2.3 (bottom right) depicts the heterogeneous nature of the reservoir. Two high-permeable streaks form short-cuts from injector to producer. Water injection with a constant pressure in the injector, and a constant, but somewhat lower, pressure in the producer will therefore not result in a desirable flat displacement front between water and oil. Instead, ‘fingering’ will occur in the high-permeability zones, and shortly after the start of water injection the oil production wells will begin to produce an
increasing fraction of water. Moreover, after injecting a large amount of water, there will still remain oil in the poorly swept area mid-in between the two streaks. Although we cannot completely counteract this effect using ICVs we can to a certain extent redirect the flow as indicated in the seven snapshots in Figure 2.3. By shutting-in injector valves that are close to one of the streaks and producer valves close to the other streak, the streamlines from injector to producer are rotated almost 90 degrees. As a result, the remaining oil in-between the streaks is now swept, and recovery for a given amount of water injected is significantly increased. We note that this result was obtained with wells operating under pressure constraints, i.e. closing of ICVs, either entirely or partially, has to be accompanied by a reduction in total well flow rates to avoid exceeding maximum or minimum pressures in the injector or producer respectively. Experience with larger, more realistic, reservoir models indicates that the theoretical scope for recovery increase using model-based optimization is typically in the order of zero to five percent, with similar economic benefits. Larger benefits can possibly be expected when these techniques are applied to reduce the amount of chemicals used for enhanced oil recovery during tertiary recovery.

2.6 Robust optimization

2.6.1 Multiple realizations

One of the essential aspects of model-based optimization of subsurface flow is the very large uncertainty of the model parameters as a result of limited knowledge of the underlying geology. One of the ways to cope with this uncertainty is to use multiple subsurface models, also known as an ensemble of geological realizations. Van Essen et al. (2009) implemented a robust ensemble-based optimization strategy to maximize the expected value \( E \) (i.e. the ensemble mean) of an objective function \( J_{RO} \) according to

\[
\max_{u_{i,k}} J_{RO} = \max_{u_{i,k}} E_\theta \left[ J \left( u_{i,k} \cdot y_{i,k}^{\text{TR}} \left( u_{i,k} \right), \theta^i \right) \right] \approx \max_{u_{i,k}} \frac{1}{R} \sum_{i=1}^{R} J \left( u_{i,k} \cdot y_{i,k}^i \left( u_{i,k} \right), \theta^i \right),
\]

(2.83)

where \( \theta^i \) and \( y^i \) are the model parameter and output vectors of realizations \( i = 1, \ldots, R \). The authors compared three flooding methods as applied to the hundred realizations: 1) an often used reactive water flooding control strategy, where the production wells are shut-in once the water-oil ratio exceeds a preset maximum, 2) nominal optimization strategies (100 in total) based on the individual realizations, and 3) the robust strategy based on maximizing the expectation as per equation (2.83). In addition the authors applied the same robust strategy to a different set of 100 realizations drawn from the same population of reservoir models, to confirm the robustness of the strategy. The results of the comparison clearly show the value of optimization compared to reactive control, and the additional benefit of a robust optimization strategy: the mean recovery of the 100 simulations is clearly the highest for the robust strategy; see Figure 2.4. The price to pay is the need to perform forward and adjoint simulations for each realization during every iteration step in the optimization procedure.
2.6.2 Risk handling in robust optimization

In Figure 2.4 it can be observed that not only the mean of the robust objective function values has increased compared to the reactive approach, but that, as a by-product of the optimization, also the variance has been reduced. This suggests the possibility to use a bi-objective optimization approach which attempts to intentionally increase the mean and reduce the variance. However, this will then usually lead to a trade-off between the two objectives: both minimizing the variance and maximizing the mean will result in a lower increase in the mean value than when only maximizing the mean and in a lower decrease in the variance than when only minimizing the variance. In its simplest form, one can use an objective function defined as (Van Essen, 2015)

\[
J_{RO} = E_0 \left[ J(u, y, \theta) \right] - \gamma Var_i \left[ J(u, y, \theta) \right]
= E_0 \left[ J(u, y, \theta) \right] - \gamma E_0 \left( J(u, y, \theta) - E_0 \left[ J(u, y, \theta) \right] \right)^2,
\]

where \( \gamma \) is a weight factor with a value between zero and one, and where we have left out the indices for discrete time and realization number (c.f. equation (2.83)). Van Essen (2015) describes how this risk-balanced robust objective function can be implemented in an adjoint-based framework. Earlier implementations, not using an adjoint, can be found in e.g., Yeten et al. (2003), Bailey et al. (2005) and Yasari et al. (2013).

In Capolei et al. (2015), a similar mean-variance bi-objective optimization approach has been implemented where the choice between the two objectives is based on risk theory as used in the financial community (portfolio optimization). A more general bi-objective approach gives the decision maker a full range of choices in the form of a Pareto front, i.e. a convex curve in the two-dimensional mean-variance space which provides the option to increase the mean at the cost of the variance or vice-versa. Liu and Reynolds (2016a, 2016b) have investigated several methods to compute the Pareto front for water flooding optimization using adjoint-based techniques.

One of the main limitations of the mean-variance optimization approach is the symmetric nature of the variance which also penalizes the best cases, while generally, in a maximization problem, the decision maker is mainly concerned with the lower tail of the objective function.
distribution. A risk-averse decision maker prefers to avoid negative outcomes more than to seek for positive ones. In Siraj et al. (2015) various asymmetric risk measures have been studied and implemented. Moreover, the authors study the effect of economic uncertainties (oil price fluctuations) in addition to geological uncertainties.

2.6.3 Constraint handling in robust optimization

A difficulty in the simulation of multiple subsurface models with a given production strategy may occur when output constraints are present because it will usually be impossible to find a single strategy that honours the constraints in all realizations (or, if it is possible to find one, it will be likely be suboptimal compared to a reactive strategy). Moreover, in applying the optimal strategy to a real field, the real rock and fluid properties will never be exactly equal to those used in any of the model realizations, and therefore the optimal strategy will encounter different constraints. A pragmatic solution to cope with this problem is as follows:

1) Run forward simulations for all ensemble members using a single strategy (with rate controls, pressure controls or a mixture of them) that honours the input (bound) constraints. Let the simulator deal with the output constraints, i.e. use a reactive control approach in which the simulator switches from rate control to pressure control, or vice-versa, when an output constraint is reached. After running the backward (adjoint) simulation for all ensemble members this will then result in gradients with respect to either pressures or rates switching at different moments in time for each realization.

2) Repeat a single forward simulation for each ensemble member using either rates or pressures. Because one can use the converged forward results from step 1) this is a relatively quick procedure in which no Newton iterations are required.

3) Continue the optimization with either rate controls and gradients with respect to rates, or pressure controls and gradients with respect to pressures. After the next set of forward runs, constraint switching may have occurred again which can then be dealt with again as described under 2).

An alternative method to cope with control switching, using gradient projection, has been described by Krogstad et al. (2016).

2.7 Implementation aspects*

2.7.1 Automatic differentiation*

Although the adjoint method is computationally very efficient, it has some serious drawbacks: 1) it requires access to the reservoir simulation code, 2) coding the necessary partial derivatives represents a major programming effort, and 3) whenever new physics is implemented in the simulation code, the corresponding adjoint information needs to be programmed in as well. A large step forward to eliminate drawbacks nrs. 2) and 3) has been made over the past decade through the development of automatic differentiation techniques. These make use of variable types that consist of a scalar or vector together with its derivative values with respect to a predefined set of independent variables. In combination with a set of algorithmic rules to perform differentiation operations on basic algebraic combinations of variables, they allow for a relatively simple way to implement and maintain the adjoint method. Younis and Aziz (2009) and Krogstad et al. (2015) describe implementations of
2.7.2 Memory requirements*

To compute an accurate numerical derivative with the adjoint method it is necessary that all steps taken in the forward simulation are exactly mirrored in the backward simulation. This implies that the states and their partial derivatives (Jacobians) should be computed at exactly the same moments in time. An important element of the adjoint solution is formed by computing the Jacobians $\frac{\partial g_i}{\partial u_k}$ at every time step. These Jacobians are already computed during the forward simulation where they are required for the Newton iteration process, and one way to re-use them for the adjoint simulations is therefore to store them on the way forward. However, this leads to a very high memory requirement, or, when the storage is done on disk, to a high amount of input/output. Alternatively, one can recompute the Jacobians on the way back which reduces the memory requirements but is obviously a more time-consuming solution. Note that the states always need to be stored because they cannot be recomputed on the way back without state information from one time step before which is not available during the backward simulation.

2.7.3 Automatic time stepping*

Another implementation aspect concerns the automatic time stepping that is used in most reservoir simulators. If during the iterative optimization process different sets of time steps are used for the iteration steps (or for the line search steps) this may result in small but noticeable differences in the state variables and the Jacobians. Especially when the optimum is being approached (i.e. when the gradient becomes very ‘flat’), this may lead to erroneous results and even failure to converge. Strategies to temporarily ‘freeze’ the time step sets during optimization and/or line searching may therefore be required; see Volkov and Voskov (2016).

2.7.4 Solving the adjoint linear system*

Another numerical implementation aspect concerns the solution of the linear systems of equations that emerge in the backward simulations. As discussed in Section 2.3.3, these are the transpose of the tangent linear model, i.e. of the linearized system equations of the forward model which are a key element in the Newton iteration process to solve the nonlinear system equations. The efficient solution of the forward linear system equations constitutes a dedicated branch of research and several very powerful strategies have been developed. In particular most reservoir simulators use a technique called constrained pressure residual (CPR) preconditioning in which the pressure equation is ‘condensed out’ from the coupled pressure-saturation equation (or the pressure-component accumulation equation). This significantly improves its condition number and therefore speeds up its solution which is then subsequently used as a starting point for solving the coupled equation. Efficiently solving the adjoint equations, i.e. the transpose of the forward equations, therefore requires transposing the matrices in the CPR procedure and performing all steps in reversed order. A detailed description of this adjoint CPR procedure has been given by Han et al. (2013).

2.8 Auxiliary topics*

2.8.1 Bang-bang control*

Sudaryanto and Yortsos (2000, 2001) observed that sometimes the iteration process during flooding optimization leads to controls that have reached either their maximum or their
minimum allowed values. Zandvliet et al. (2007) analyzed under which conditions such bang-bang controls can occur, and showed that this is often the case if the problem is linear in the controls, i.e. if the influence of \( \mathbf{u} \) in equations (2.1) and (2.3) can be expressed as \( \mathbf{x}_k(\mathbf{u}_k) = \mathbf{B}_k \mathbf{x}_k \) and \( \mathbf{y}_k(\mathbf{u}_k) = \mathbf{D}_k \mathbf{u}_k \) with \( \mathbf{B} \in \mathbb{R}^{n \times m} \) and \( \mathbf{D} \in \mathbb{R}^{p \times m} \) time-invariant matrices, while at the same time the integral term in the objective function is linear in the controls, and the constraints are limited to input bounds as expressed in equation (2.64). If it is known in advance that the optimal solution is of a bang-bang nature, the iterative optimization procedure can be simplified by searching for the switching moments for the inputs only, instead of for the entire input trajectory. A practical advantage of bang-bang control is the possibility to implement the inputs with on-off control valves which are considerably cheaper than continuously-variable valves.

### 2.8.2 Multi-scale optimization

The total number of control variables in optimization problem (2.8) is equal to \( n \times K \) where \( n \) is the number of elements in the input vector \( \mathbf{u} \) and \( K \) is the number of simulation time steps. With up to hundreds of wells and hundreds of time steps for realistically-sized problems the total number of control variables may therefore be in the order of \( 10^3 \) to \( 10^4 \). Experience shows that often the optimal trajectories, as found in an iterative procedure, display a ‘nervous’ character with rapid fluctuations around a slowly changing average. Moreover, it is found that these fine-scale time fluctuations of the optimal trajectories have little or no influence on the objective function value, which implies that there is scope to simplify or regularize the input trajectories. A straightforward way to do so is to use control intervals of predefined length which are spanning many time steps, see e.g. Sarma et al. (2005) or Kraaijevanger et al. (2007). An alternative method was proposed by Lien et al. (2008) in which the problem is initially solved for just two control intervals in time and two groups of controls in space, i.e. for a total number of four generalized input variables corresponding to a control super-vector \( \mathbf{u} \) with four elements only. Using a sensitivity measure of the objective function to changes in each of the four elements, a gradual refinement of the control vector, both in time and space is obtained, eventually leading to smooth optimal input trajectories and sometimes also a faster convergence of the optimization algorithm.

### 2.8.3 Augmented Lagrangian

Doublet et al. (2009) described the use of an alternative definition of the modified objective function, known as an augmented Lagrangian formulation, according to

\[
\tilde{J}(\mathbf{u}_{1,K},\mathbf{x}_{0,K},\mathbf{y}_{1,K},\lambda_{0,K},\mathbf{u}_{1,K},\mathbf{v}_{1,K}) \leq \tilde{J}(\mathbf{u}_{1,K},\mathbf{x}_{0,K},\mathbf{y}_{1,K},\lambda_{0,K},\mathbf{u}_{1,K},\mathbf{v}_{1,K})
\]

\[
+ \frac{\theta}{2} \sum_{k=1}^{K} \mathbf{g}_k^T(\mathbf{u}_k,\mathbf{x}_{k-1},\mathbf{x}_k) \mathbf{g}_k(\mathbf{u}_k,\mathbf{x}_{k-1},\mathbf{x}_k),
\]

where \( \theta \) is a positive scalar, referred to as the penalty parameter, and where \( \mathbf{g}_k \) is defined by system equation (2.1). The effect of the additional quadratic terms \( \mathbf{g}_k^T \mathbf{g}_k \) is meant to make the optimization procedure less sensitive to errors (residuals) in the ‘forward’ solution of equation (2.1). Such errors sometimes lead to computational problems during the ‘backward’ solution of the Lagrange multiplier equation (2.20); see Vakili et al. (2005). Moreover, reducing the sensitivity to errors allows for a reduced tolerance on the Newton-Raphson iterations during the forward solution which may lead to computational gains. As noted in Section 0, the use of an augmented Lagrangian formulation can also be applied to implement state or input constraints; see Chen et al. (2010).
2.8.4 Continuous versus discrete adjoint *
In various areas of engineering discussions have been held about the relative merits of first discretizing the forward flow equations in time and thereafter deriving the discrete-time adjoint equations (the first-discretize-then-differentiate approach), versus first deriving the continuous-time adjoint equations and then discretizing the forward and adjoint equations in time (the first-differentiate-then-discretize approach). Most authors seem to agree that both methods can be applied as long as the forward and backward equations are truly each other’s adjoint, which implies discretization at identical moments in time of the forward and backward equations using identical discretization schemes. Brouwer (2004) discussed these aspects, and recently Kourounis et al. (2014) re-evaluated the matter in relation to adjoints for multi-component (compositional) simulation. Currently available large-scale reservoir simulation packages all follow the first-discretize-then-differentiate approach.

2.8.5 Reduced-order modeling *
Although typical reservoir models contain $10^4$ to $10^6$ state variables (grid block pressures and saturations) the extent to which these individual states can be influenced through changing the controls $u$ is very limited. It can be shown that only a very small number of spatial patterns, in terms of pressures or saturations, are controllable, and consequently, the amount of control that can be exerted on the oil-water front is also very limited; see e.g. Fyrozaee and Yortsos (2006), Ramakrishnan (2007), Zandvliet et al. (2008), Jansen et al. (2009) and Van Doren et al. (2013). Therefore, the dynamics of a high-order reservoir flow model, if controlled by only a limited number of inputs, can be captured in a low-order model in terms of a small number of generalized state variables $z$. An efficient way to empirically derive such a low-order model is with the aid of a technique called proper orthogonal decomposition (POD), also known as principal component analysis, Karhuenen-Loève decomposition, the method of empirical eigen functions, or model order reduction. For applications of POD and related methods to flow through porous media see e.g. Vermeulen et al. (2004), Hein et al. (2004), Van Doren et al. (2006), Markovinović et al. (2006), Cardoso (2009, 2010a, 2010b), Krogstad (2011), Gildin et al. (2013) and Rousset et al. (2014). In the POD method the generalized state variables $z$ are related to the original state variables $x$ according to

$$x = \Phi z,$$  \hspace{1cm} (2.86)

where $\Phi = [\varphi_1, \varphi_2, \ldots, \varphi_i] \in \mathbb{R}^{d \times \ell}$ is a transformation matrix with $\ell << n$. The basis functions $\varphi_i$ are the eigenvectors of a matrix $XX^T$ which is a low-rank approximation of the spatial covariance between the state variables over time:

$$\text{cov}(x_{i,\ell}) \approx XX^T = \begin{bmatrix} (x_1 - \bar{x})^T \\ (x_2 - \bar{x})^T \\ \vdots \\ (x_\ell - \bar{x})^T \end{bmatrix},$$  \hspace{1cm} (2.87)

where $\bar{x} = \Sigma_{i=1}^\ell x_i / \ell$ is the time-averaged state. Note that because of the rank deficiency of $XX^T$ it is sufficient to solve the eigen value problem for the much smaller matrix $X^T X$. The theoretical significance of the POD reduction method is that the matrix of basis functions $\Phi$ can be interpreted as a low-rank approximation to the controllability Gramian of the
Gradient-based optimization of flow through porous media, version 3, September 2016

(Linearized) system equations, see e.g. Antoulas (2005). Controllability Gramians can be used to quantify the (limited) extent to which it is possible to influence the states by changing the controls (Zandvliet et al., 2008, Van Doren et al., 2013). An attempt to benefit from the limited controllability, and from the associated low-order dynamics of the controlled system, to speed-up the flooding optimization procedure was presented by van Doren et al. (2006). They used a nested approach with an inner loop to perform the optimization in reduced-order space and outer loop to correct the approximate results from the inner loop using the high-order model. The speed-up in the inner loop results from replacing the high-order system of equations (2.1) by a low-order equivalent,

$$g_k(u_k, \Phi z_{k-1}, \Phi z_k) = 0, \quad k = 1, 2, ..., K,$$

which strongly reduces the size of the underlying linear systems of equations that need to be solved during the Newton-Raphson iterations in the forward simulation. Although the number of state variables was reduced drastically (from 4050 to between 20 and 100), the computational gain was limited (about 35%) because of the nonlinear nature of the problem. Nevertheless the results demonstrated that the dynamics of flooding optimization with a fixed well configuration is indeed governed by a low-order set of equations. If and how this theoretical result can be used to achieve practically relevant computational speed-ups remains a matter of further research. Promising results were recently obtained in Krogstad et al. (2009) and Cardoso et al. (2010).

2.9 References for Chapter 2

Antoulas, A.C., 2005: Approximation of large-scale dynamical systems, SIAM, Philadelphia.


Bryson, A.E. and Ho, Y-C., 1975: Applied optimal control, Taylor and Francis (Hemisphere), Levittown.


Han, C., Wallis, J., Sarma, P. et al., 2013: Adaptation of the CPR preconditioner for efficient solution of the adjoint equation. SPE Journal 18 (2) 207-213. https://doi.org/10.2118/141300-PA


Liu, W., Ramirez, W.F. and Qi, Y.F., 1993: Optimal control of steam flooding. SPE Advanced Technology Series 1 (2) 73-82. https://doi.org/10.2118/21619-PA


Van Doren, J.F.M., Markovinović, R. and Jansen, J.D., 2006: Reduced-order optimal control of water flooding using POD. *Computational Geosciences* **10** (1) 137-158. [https://doi.org/10.1007/s10596-005-9014-2](https://doi.org/10.1007/s10596-005-9014-2)


Younis, R. and Aziz, K., 2009: Parallel automatically differentiable data-types for next-generation simulator development. Paper SPE106493 presented at the *SPE Reservoir*
Simulation Symposium, Houston, U.S.A., 26-28 February. https://doi.org/10.2118/106493-MS


3 Towards operational use

3.1 Reservoir surveillance and computer-assisted history matching

Within the exploration and production life cycle of an oil field various phases can be distinguished. The use of adjoint-based flooding optimization is particularly relevant to the field development phase, during which decisions are taken about the position and the number of wells and about the capacities of the surface facilities for processing of the produced and injected fluids (Sarma and Chen, 2008b; Van Essen et al., 2010). During the operational phase, surveillance of reservoir performance is usually performed by measuring pressures at the well head daily, and production rates of oil, gas and water in the individual wells more infrequently, say monthly. All these measurements are typically used in tabular form or displayed graphically to get an impression of the state of the reservoir. Moreover, they form the input for specialized semi-analytical techniques such as material balance analysis or decline curve analysis to extrapolate past well performance with the aim to predict future performance on a time horizon of weeks to months. However, large-scale numerical reservoir simulation traditionally plays no role during surveillance, and only appears on the stage again after many years, say five to ten, when it is necessary to perform a ‘field redevelopment’ study, which often involves building a set of entirely new geological and reservoir simulation models. Calibration of these new models is usually performed using historic production data. This history matching process involves adapting uncertain model parameters, such as permeabilities or porosities, until the simulated well rates and pressures match their measured counterparts in some averaged sense. Traditionally performed manually, history matching is becoming more and more dependent on numerical procedures, known as computer-assisted history matching, and a vast number of methods have been proposed over the past decades; see e.g. Oliver et al. (2008) or Evensen (2009) for recent overviews. Parameter estimation through minimizing the mismatch between measured and simulated production data was in fact the first area of application for adjoint-based methods in the petroleum industry; see e.g. Chen et al. (1974), Chavent et al. (1975), Li et al. (2003) and Oliver et al. (2008). More recently, ensemble Kalman filters and smoothers, streamline-based methods, genetic algorithms and other data assimilation techniques have become serious competitors.

3.2 Closed-loop reservoir management

3.2.1 Combined flooding optimization and history matching

Over the past decade a gradually increasing number of studies have proposed to combine computer-assisted history matching and flooding optimization to keep reservoir simulation models evergreen such that they can also be used during the operational phase of oil field development, and not just during field development planning; see Figure 3.1.
The idea for closed-loop optimization or closed-loop reservoir management (CLRM) has been around for many years in different forms, often centered around attempts to improve reservoir characterization from a geosciences perspective. Moreover, recently closed-loop or real-time approaches to hydrocarbon production have received growing attention as part of various industry initiatives with names as smart fields, i-fields, e-fields, self-learning reservoir management or integrated operations; see Jansen et al. (2005, 2008, 2009) for references. However, whereas the focus of most of these initiatives is primarily on optimization of short-term production, CLRM is more focused on life-cycle optimization, i.e. on processes at a timescale from years to tens of years. Moreover, in contrast to the geosciences-focused approach, we emphasize the need to focus on those elements of the modeling process that can both be verified from measurements and bear relevance to controllable parameters such as well locations or, in particular, production parameter settings. The underlying hypothesis is that

“It will be possible to significantly increase life-cycle value by changing reservoir management from a periodic to a near-continuous model-based controlled activity.”

We stress that, in our view, “closed-loop” does not imply removal of human judgment from the loop. The use of model-based optimization and data assimilation techniques should result in a reduction of time spent on repetitive and tedious human activities and thus in more time that may be spent on judging results and taking decisions. We will not address CLRM in any further detail in these notes, and refer to Jansen et al. (2009), and Peters et al. (2010) for recent detailed information. Although the particular history matching and flooding optimization techniques used in CLRM are not of prime importance, we have, in line with the topic of these notes, listed several closed-loop studies that have used adjoint-based flooding optimization; see Brouwer et al. (2004), Overbeek et al. (2004), Jansen et al. (2005, 2008, 2009), Sarma et al. (2005, 2006, 2008), Naevdal et al. (2006), Wang et al. (2009), Chen et al.
An alternative approach to cope with the problem of uncertainty in reservoir parameters is to optimize a reactive (feed-back) strategy. A recent application along these lines is described by Dilib and Jackson (2013) and Dilib et al. (2015) who refer to their approach as ‘closed-loop feedback control’. In this case the expression ‘closed-loop’ refers to the use of output information (pressures, phase rates) to immediately change the input, without using an intermediate data assimilation step to update the reservoir model. Although the authors did not use adjoint-based optimization, the same approach could be used with adjoints.

### 3.3 Hierarchical optimization

Section 2.8.2 addressed some problems that may occur in realistically-sized flooding optimization studies because of the very large number of control variables: 1) rapidly fluctuating controls, and 2) different combinations of controls that result in nearly identical objective function values. The latter implies that there is most likely redundancy in the controls. A related issue was highlighted by Van Essen et al. (2011). They showed that an optimal trajectory may result in increased recovery or financial benefits at the end of the optimization period, but sometimes at the cost of reduced short-term benefits, at least seemingly so. Such reduced short-term gains will be a major obstacle to implementation of the optimal trajectory in practice because it will be very difficult to convince the production-oriented part of an organization to reduce a certain short-term income in favor of a much more uncertain long-term benefit. Fortunately, it turns out that the drop in short-term gains is often not an essential element of the optimal strategy but rather a result of the redundancy in the controls in combination with a long-term objective. Van Essen et al. (2011) demonstrated that it is possible to first optimize the long-term objective and thereafter, using the redundancy in the controls, a second, short-term objective. Formally this can be done by determining the Hessian of the objective function along the optimal trajectory for the primary (i.e. long-term) objective. This gives the direction in which it is possible to maximize a secondary (short-term) objective without changing the primary objective’s value. A somewhat more ad-hoc, but computationally much more efficient method to achieve the same result is to first optimize the primary objective, and thereafter alternatingly the secondary and the primary objective until convergence.

As an alternative to optimizing long-term and short-term gains hierarchically, it is also possible to treat them as elements in a bi-objective optimization problem, similar to what was described in Section 2.6.2 to balance maximizing the mean and minimizing the variance in robust optimization. This was done by, e.g. Liu and Reynolds (2016).

### 3.4 Multi-level optimization

Other reasons why CLRM, i.e. combined model-based flooding optimization and computer-assisted history matching may be difficult to implement in practice are 1) the time-consuming nature of the history matching process, 2) the poor representation of near-well bore reservoir dynamics by reservoir simulation models (because of discretization errors), 3) the very large uncertainty in reservoir parameters (even when using frequent model updating, and even when somewhat mitigating the uncertainty with the aid of robust optimization), and 4) the cultural differences between the reservoir engineering and production engineering disciplines.
Similar problems occur in the process industry and the standard solution is to use a multi-level control structure in which the results of a higher layer serve as optimal reference for the next lower layer. A similar structure has been proposed by Saputelli et al. (2006) for use in flooding optimization, but a systematic implementation of multi-level flooding optimization has not yet been reported. Van Essen et al. (2012) proposed a two-level control structure where the upper level is formed by an adjoint-based life-cycle optimization algorithm. The lower level is formed by a data-driven (black-box) response model which has a limited prediction horizon but runs very fast and can be used in an operational environment. The lower-level algorithm attempts to track the optimal output of the upper-level algorithm by manipulation of the inputs, and thus acts as a ‘disturbance rejection’ technique that compensates for the errors in the upper-level model. Outstanding questions include the best choice for the lower-level algorithm, and workflow aspects, and we expect that the development of practically applicable multi-level optimization techniques will be important to extend the use of flooding optimization from field development planning into the operational domain.

3.5 Other applications

For completeness sake, we will list some other applications of adjoint-based flooding optimization. This concerns primarily the optimization of well locations and well trajectories, which is an important aspect of field development planning. Many algorithms have been proposed to solve the well location optimization problem of which the adjoint-based ones form only a small subset; see Wang and Reynolds (2007), Zandvliet et al (2008), Sarma and Chen (2008a) and Vlemmix et al. (2009). Moreover, similar flooding optimization problems as described in these notes for oil reservoir flow occur in groundwater flow, where the objective is to clean up pollution through prolonged flushing with water, possibly in combination with surfactants or other cleaning agents. However, nearly all of the groundwater optimization references describe adjoint-free optimization methods with the exception of two papers by Merckx (1991a, 1991b).

3.6 Alternative optimization methods

We conclude by noting that adjoint-based techniques are certainly not the only option for flooding optimization problems. Although they are extremely efficient, their implementation has major drawbacks as was discussed briefly in Section 2.7.1. Therefore various alternative flooding optimization methods have been proposed, of which we mention, without being complete, genetic algorithms and simulated annealing (Yang et al., 2003), streamline-based methods (Thiele and Batycky, 2006; Alhuthali et al. 2007, 2008, 2009), and ensemble optimization (EnOpt) (Chen, 2008 and Chen et al., 2009, Do and Reynolds, 2013). Especially the EnOpt method has seen a major increase in popularity over the past years.

3.7 References for Chapter 4


Bryson, A.E. and Ho, Y.-C., 1975: Applied optimal control, Taylor and Francis (Hemisphere), Levittown.


Nomenclature

Notes:
- Some symbols occur more than once because they have a different meaning in different parts of the text.
- Some symbols have different dimensions (an corresponding units) in different parts of the text.
- The dimensions of vectors and matrices have only been indicated when all elements have the same dimensions.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Dimensions</th>
<th>SI units</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>column of linear constraint matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>linear constraint matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>A</td>
<td>Jacobian</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>discount factor</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>vector of constants in linear constraints</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>input matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>Jacobian</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>equality constraint</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>c</td>
<td>vector of equality constraints</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>subset of $\mathbb{R}^q$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>distribution covariance matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C_{ej}</td>
<td>ensemble cross-covariance vector</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C_{en}</td>
<td>ensemble covariance matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>Jacobian</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>d</td>
<td>inequality constraint</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>d</td>
<td>vector of inequality constraints</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>subset of $\mathbb{R}^r$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>direct throughput matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>Jacobian</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>domain</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>expectation operator</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>number of ensemble members in EnOpt</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>f</td>
<td>nonlinear vector-valued (system) function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>Jacobian</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>search direction</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>nonlinear vector-valued (system) function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>nonlinear super vector-valued (system) function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>h</td>
<td>nonlinear vector-valued (output) function</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>Hessian matrix</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>Hamiltonian</td>
<td>-, M</td>
<td>-, $$</td>
</tr>
<tr>
<td>i</td>
<td>counter</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j</td>
<td>counter</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( j )</td>
<td>nonlinear vector-valued (output) function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( j )</td>
<td>nonlinear super vector-valued (output) function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{I} )</td>
<td>identity matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( J )</td>
<td>objective function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \overline{J} )</td>
<td>average objective function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \widetilde{J} )</td>
<td>modified objective function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{J} )</td>
<td>dual objective function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nabla J )</td>
<td>gradient vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nabla J' )</td>
<td>regularized gradient vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nabla J'' )</td>
<td>regularized gradient vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \nabla J''' )</td>
<td>regularized gradient vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( k )</td>
<td>discrete time (time step counter)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( K )</td>
<td>total number of time steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \ell )</td>
<td>number of basis functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathcal{L} )</td>
<td>Lagrangian</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( m )</td>
<td>number of elements in input vector ( \mathbf{u} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M )</td>
<td>number of elements in super input vector ( \mathbf{U} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>number of elements in state vector ( \mathbf{x} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( N )</td>
<td>number (general)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( N )</td>
<td>number of elements in super state vector ( \mathbf{x} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td>number of elements in output vector ( \mathbf{y} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td>pressure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{p} )</td>
<td>pressure vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P )</td>
<td>number of elements in super output vector ( \mathbf{Y} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{P}^\perp )</td>
<td>complementary orthogonal-projection matrix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q )</td>
<td>number of equality constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q )</td>
<td>flow rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{q} )</td>
<td>vector of flow rates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r )</td>
<td>number of inequality constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r )</td>
<td>cost or revenue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{r} )</td>
<td>residual vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( R )</td>
<td>number of geological realizations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s )</td>
<td>vector of slack variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathcal{S} )</td>
<td>set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( t )</td>
<td>time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( u )</td>
<td>input variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{u} )</td>
<td>input vector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{u} )</td>
<td>ensemble mean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{\bar{u}} )</td>
<td>distribution mean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{u} )</td>
<td>super vector of input variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( U )</td>
<td>subset of ( \mathbb{R}^m )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{U} )</td>
<td>matrix of ensemble mean-shifted control vectors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{\bar{U}} )</td>
<td>matrix of ensemble mean-shifted control vect.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
$v$ element of unit tangent vector
$\mathbf{v}$ unit tangent vector
$\mathbf{V}$ matrix of unit tangent vectors
$x$ state variable
$\mathbf{x}$ state vector
$\mathbf{x}$ super vector of state variables
$X$ subset of $\mathbb{R}^n$
$\mathbf{X}$ snapshot matrix
$y$ output variable
$\mathbf{y}$ output vector
$\mathbf{y}$ super vector of output variables
$Y$ subset of $\mathbb{R}^n$
$z$ transformed state vector
$\alpha$ dimensionless valve opening
$\mathbf{\alpha}$ vector of dimensionless valve openings
$\beta$ interpolation variable
$\gamma$ step size
$\gamma$ weight factor
$\delta_k$ Kronecker delta
$\Delta t$ time step
$\epsilon$ small number
$\epsilon$ error
$\theta$ penalty parameter
$\mathbf{\theta}$ parameter vector
$\lambda$ Lagrange multiplier
$\mathbf{\lambda}$ vector of Lagrange multipliers
$\mathbf{\lambda}$ super vector of Lagrange multipliers
$\mu$ Lagrange multiplier
$\mathbf{\mu}$ vector of Lagrange multipliers
$\mathbf{\mu}$ super vector of Lagrange multipliers
$\nu$ Lagrange multiplier for equality constraint
$\mathbf{\nu}$ vector of Lagrange multipliers for eq. constr.
$\tau$ reference time interval for discounting
$\tau$ time of flight along a streamline
$\phi$ column of reduction matrix $\Phi$
$\Phi$ reduction matrix
$\omega$ Lagrange multiplier for inequality constraint
$\omega$ vector of Lagrange multipliers for ineq. constr.

Subscripts
$0$ initial
$a$ augmented
$g$ gas
$\text{inj}$ injector
\begin{itemize}
\item[$k$] discrete time
\item[$l$] liquid
\item[$o$] oil
\item[$\text{prod}$] producer
\item[$RO$] robust optimization
\item[$t$] total
\item[$tf$] flowing tubing head
\item[$w$] water
\item[$wi$] injected water
\item[$wp$] produced water
\item[$wf$] flowing well bore
\end{itemize}

\textbf{Superscripts}
\begin{itemize}
\item[0] end point saturation
\item[$T$] transpose
\end{itemize}
**Glossary**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD-GPRS</td>
<td>Automatic Differentiation General Purpose Research Simulator</td>
</tr>
<tr>
<td>AD-MRST</td>
<td>Automatic Differentiation Matlab Reservoir Simulation Toolbox</td>
</tr>
<tr>
<td>CLRM</td>
<td>Closed-Loop Reservoir Management</td>
</tr>
<tr>
<td>CPR</td>
<td>Constrained Pressure Residual</td>
</tr>
<tr>
<td>DOI</td>
<td>Digital Object Identifier</td>
</tr>
<tr>
<td>EnOpt</td>
<td>Ensemble Optimization</td>
</tr>
<tr>
<td>GRG</td>
<td>Generalized Reduced Gradient</td>
</tr>
<tr>
<td>ICV</td>
<td>Inflow Control Valve</td>
</tr>
<tr>
<td>LBFGS</td>
<td>limited-memory Broyden Fletcher Goldfarb Shanno</td>
</tr>
<tr>
<td>NPV</td>
<td>Net Present Value</td>
</tr>
<tr>
<td>POD</td>
<td>Proper Orthogonal Decomposition</td>
</tr>
<tr>
<td>SPE</td>
<td>Society of Petroleum Engineers</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
</tbody>
</table>
Index

analysis ......................................................... 54
principal component ................................... 54
arc feasible .................................................. 19
assimilation data ............................................. 63
back tracking .................................................. 29
balance material ............................................. 63
cash flow ...................................................... 33
complementarity ............................................. 25
strict ........................................................... 25
condition complementarity ................................ 25
Karush-Kuhn-Tucker ........................................ 25
necessary ...................................................... 8, 19, 24
optimality ...................................................... 8, 19
sufficient ...................................................... 8
 constrained pressure residual ........................... 52
constraint ...................................................... 33, 35, 42, 51
active .......................................................... 22
bound ............................................................ 43
equality ....................................................... 11, 18, 34, 42
inactive ........................................................ 22
inequality .................................................... 22, 34, 42
input ............................................................ 43
output .......................................................... 42
state ............................................................. 42
state-path ..................................................... 42
strongly active ............................................. 24
weakly active ............................................... 24
control bang-bang ........................................... 53
multi-level ..................................................... 66
optimal ........................................................ 34
reactive ........................................................ 49
convexity ..................................................... 10
curvature ...................................................... 10, 28
curve decline ............................................... 63
decomposition Karhuenen-Loève .................. 54
proper orthogonal ........................................ 54
delta Kronecker ............................................. 35
derivative directional ...................................... 15, 20
differentiation automatic .................................. 51
implicit ......................................................... 41
implicit ......................................................... 16
direction feasible ........................................... 18
search .......................................................... 26
dual ............................................................. 39
ensemble ...................................................... 49
epigraph ........................................................ 10
equation Euler-Lagrange .................................. 20, 25, 36, 44
output ........................................................ 32
system ........................................................ 32
estimation parameter ....................................... 63
evaluation function ......................................... 27
upper ........................................................... 8
field smart ..................................................... 64
form residual .................................................. 32
function convex .............................................. 10
cost ............................................................ 8
Dirac delta .................................................... 35
modified objective ......................................... 14
multivariate ................................................... 9
objective ..................................................... 8, 33
penalty ......................................................... 47
performance .................................................. 8
univariate ..................................................... 8
functional ..................................................... 12
gradient ....................................................... 7, 9, 26, 28
generalized reduced ....................................... 46
projected ....................................................... 15
reduced ......................................................... 46
gravity ........................................................ 32
Hamiltonian ................................................... 38
handling external constraint ................................ 43
internal constraint .......................................... 44
Hessian ......................................................... 9, 28, 65
history matching ............................................ 63
computer-assisted ........................................ 31, 63
manual ........................................................ 63
input controllable ........................................... 32
iteration Newton-Raphson ................................ 28
Karush-Kuhn-Tucker ...................................... 25
Kuhn-Tucker ................................................ 25
Lagrangian ................................................... 14, 37
augmented ................................................... 47, 53
line search .................................................... 29
location well ................................................. 66
lumping constraint .......................................... 46
management closed-loop reservoir ..................... 64
real-time reservoir ........................................ 64
self-learning reservoir ................................... 64
matrix Hessian ............................................... 9
transformation ............................................. 54
maximum ..................................................... 8
method
Gradient rate projection programming operator
Shanno penalty quasi-Newton steepest ascent steepest descent
Zoutendijk’s minimum model
evergreen multiplier
Lagrange operation integrated optimum
operator global local optimization bi-objective closed-loop ensemble
flooding gradient-based gradient-free hierarchical life-cycle multi-level
multi-scale numerical production robust well control well location well-trajectory
parameter penalty perturbation binding non-binding point
critical feasible inflection optimal saddle stationary
convex programming nonlinear projection gradient qualification constraint
erate discount ratio
water-oil realization reduction model order regularization replacement
voidage residual risk saddle seismic
time-lapse sensitivity backward forward set
convex feasible simulation compositional slack smart field smart well strategy
nominal reactive robust surveillance time
trajectory well trust region uncertainty economic geological value
present valve control on-off.
variable input manipulated output slack state variation admissible
vector concatenated control decision derivative gradient input output state